



NAVAL POSTGRADUATE SCHOOL

Monterey, California



THESIS

J2125

UTILIZATION OF A KALMAN OBSERVER WITH
LARGE SPACE STRUCTURES

by

Bruce M. Jackson

December 1988

Thesis Advisor

Jeff B. Burl

Approved for public release; distribution is unlimited.

T241978

REPORT DOCUMENTATION PAGE

Report Security Classification Unclassified		1b Restrictive Markings		
Security Classification Authority		3 Distribution Availability of Report Approved for public release; distribution is unlimited.		
Declassification Downgrading Schedule				
Performing Organization Report Number(s)		5 Monitoring Organization Report Number(s)		
Name of Performing Organization Naval Postgraduate School	6b Office Symbol (if applicable) 33	7a Name of Monitoring Organization Naval Postgraduate School		
Address (city, state, and ZIP code) Monterey, CA 93943-5000		7b Address (city, state, and ZIP code) Monterey, CA 93943-5000		
Name of Funding Sponsoring Organization	8b Office Symbol (if applicable)	9 Procurement Instrument Identification Number		
Address (city, state, and ZIP code)		10 Source of Funding Numbers Program Element No Project No Task No Work Unit Accession No		
Title (include security classification) UTILIZATION OF A KALMAN OBSERVER WITH LARGE SPACE STRUCTURES				
Personal Author(s) Bruce M. Jackson				
Type of Report Master's Thesis	13b Time Covered From To	14 Date of Report (year, month, day) December 1988	15 Page Count 85	
Supplementary Notation The views expressed in this thesis are those of the author and do not reflect the official policy or position of the Department of Defense or the U.S. Government.				
Cosati Codes		18 Subject Terms (continue on reverse if necessary and identify by block number) Kalman Observer, large space structures.		
Id	Group	Subgroup		
Abstract (continue on reverse if necessary and identify by block number) Control of the motions and vibrations of large space structures require the knowledge of state values that may not be available due either to inability to measure the states or, the high cost of the sensors to measure the required states. One solution is the use of an observer to estimate the states from limited sensor input. The physical characteristics of large space structures and the environment they operate in will cause large amounts of noise in the measurements. The obvious observer for such an environment is the Kalman Filter which is specifically designed to produce optimal estimates in a noisy environment. A straightforward application of the Kalman Filter will be examined utilizing a steady state Kalman gain matrix. The observer performance will be examined in both matched filter/plant and reduced order filter configurations.				

Distribution Availability of Abstract Unclassified unlimited <input type="checkbox"/> same as report <input type="checkbox"/> DTIC users		21 Abstract Security Classification Unclassified	
22a Name of Responsible Individual Jeff B. Burl		22b Telephone (Include Area code) (408) 646-2390	22c Office Symbol 62BL

Approved for public release; distribution is unlimited.

Utilization of a Kalman Observer with Large Space Structures

by

Bruce M. Jackson
Lieutenant Commander, United States Navy
B.S., United States Naval Academy, 1976

Submitted in partial fulfillment of the
requirements for the degree of

MASTER OF SCIENCE IN ENGINEERING SCIENCE

from the

NAVAL POSTGRADUATE SCHOOL
December 1988

ABSTRACT

Control of the motions and vibrations of large space structures require the knowledge of state values that may not be available due either to inability to measure the states or, the high cost of the sensors to measure the required states. One solution is the use of an observer to estimate the states from limited sensor input.

The physical characteristics of large space structures and the environment they operate in will cause large amounts of noise in the measurements. The obvious observer for such an environment is the Kalman Filter which is specifically designed to produce optimal estimates in a noisy environment.

A straightforward application of the Kalman Filter will be examined utilizing a steady state Kalman gain matrix. The observer performance will be examined in both matched filter/plant and reduced order filter configurations.

50103
C.1

THESIS DISCLAIMER

The reader is cautioned that computer programs developed in this research may not have been exercised for all cases of interest. While every effort has been made, within the time available, to ensure that the programs are free of computational and logic errors, they cannot be considered validated. Any application of these programs without additional verification is at the risk of the user.

TABLE OF CONTENTS

I. INTRODUCTION	1
A. BACKGROUND	1
B. PROBLEM STATEMENT	1
C. ORGANIZATION	2
II. MATHEMATICAL MODEL	3
A. INTRODUCTION	3
B. MODAL MODEL	3
C. DISCRETE-TIME MODEL	6
III. THE OBSERVER	10
A. INTRODUCTION	10
B. KALMAN FILTER EQUATIONS	10
C. STEADY-STATE SOLUTION	12
D. OBSERVER PERFORMANCE	12
IV. SIMULATION	14
A. INTRODUCTION	14
B. PLANT AND OBSERVER DATA	14
C. SIMULATION PROGRAMS	14
D. EFFECT OF PLANT TO MEASUREMENT NOISE RATIO ON OB- SERVER PERFORMANCE	15
E. EFFECTS OF INCREASED MODES ON OBSERVER PERFORMANCE	16
F. REDUCED ORDER KALMAN OBSERVER	16
V. CONCLUSIONS AND RECOMMENDATIONS	35
A. CONCLUSIONS	35
B. RECOMENDATIONS	35
APPENDIX A. KALMAN GAIN MATRIX GENERATION PROGRAM	36

APPENDIX B. KALMAN OBSERVER AND PLANT SIMULATION	49
APPENDIX C. PROGRAM TO ESTIMATE NOISE IN KALMAN FILTER FROM UNOBSERVED MODES	62
LIST OF REFERENCES	73
INITIAL DISTRIBUTION LIST	74

LIST OF TABLES

Table 1. CUMULATIVE UNANTICIPATED NOISE FROM UNOBSERVED MODES	18
Table 2. UNANTICIPATED NOISE FROM UNOBSERVED MODES BY MODE	18

LIST OF FIGURES

Figure 1.	Observer Performance (J) PN/MN = 1.0d12	19
Figure 2.	Observer Performance (J) PN/MN = 1.0d11	20
Figure 3.	Observer Performance (J) PN/MN = 1.0d10	21
Figure 4.	Observer Performance (J) PN/MN = 5.0d09	22
Figure 5.	Observer Performance (J) PN/MN = 2.5d09	23
Figure 6.	Observer Performance (J) PN/MN = 1.0d09	24
Figure 7.	Mode 7 (Postion) Observer Performance versus PN/MN	25
Figure 8.	Settling Time versus PN/MN	26
Figure 9.	Observer Performance (J) 4 Modes (7 - 10)	27
Figure 10.	Observer Performance (J) 5 Modes (7 - 11)	28
Figure 11.	Observer Performance (J) 6 Modes (7 - 12)	29
Figure 12.	Observer Performance (J) 7 Modes (7 - 13)	30
Figure 13.	Observer Performance (J) 8 Modes (7 - 14)	31
Figure 14.	Observer Performance (J) 9 Modes (7 - 15)	32
Figure 15.	Observer Performance (J) 10 Modes (7 - 16)	33
Figure 16.	Settling Time versus number of Modes Observed	34

ACKNOWLEDGEMENTS

I would like to express my appreciation to the McDonnel Douglas Astronautics Company of Huntington Beach, California, for allowing the use of their dynamic model of a preliminary space station configuration. Also, I would like to thank Major William J. Preston, USMC for allowing the use of the mathematical model for the station dynamics which makes up the majority of Chapter II of this paper as well as portions of his simulation program. Finally, I would like to thank Professor J. B. Burl for his assistance and most of all his patience in the completion of this work.

This paper is dedicated to Christina Marie Jackson (USNA '10).

I. INTRODUCTION

A. BACKGROUND

The advent of large space structures poses a number of problems for the control engineer. Previously, the objects put into space could be treated as rigid bodies so that a single three axis sensor package could be used to tell the motion of all components. The large space structures will not be rigid, instead they will have considerable flexibility and multiple modes of vibration [Ref. 1: p. 51]

Control of the structure's attitude and vibrations requires knowing the motions of the components. One approach would be to heavily instrument the space structure, but weight and cost make this approach impractical. An alternative is to use a limited number of sensors to measure only certain states and to deduce the other required states by use of an observer algorithm.

This thesis will address the production of estimates of the states needed for control of the structure. The model used will be a early design study by McDonnell Douglas Astronautics for a dual keel space station. The techniques and problems of observation for this model are generic to all large space structures.

B. PROBLEM STATEMENT

Design of an observer for estimating the states of a large space structure breaks down into several steps. First, a mathematical model is developed for the system behavior over time. Modal analysis is used to form a system composed of decoupled second order differential equations. The use of decoupled equations allows a reduced order model to be generated by truncating the number of modal equations. A reduced order model will have all of the same mathematical qualities (and problems) but reduces the amount of time and computer resources required to do simulation.

Second, the observer is designed. The observer is designed to obtain a minimum variance estimate of the desired state values from the measurements.

Third, the observer is simulated to verify performance. Simulation runs of both a matched observer/plant system and a reduced order observer are employed. That is, the system is run where the observer is used to estimate all of the plant states and run where there are more plant states than the observer estimates.

Fourth, results are analysed and conclusions drawn based on these results. Recommendations for further areas of research are suggested based on the results and conclusions.

C. ORGANIZATION

The model of the space station is developed in Chapter II. The modal model was developed using modal analysis and discretized to form the discrete-time state equations. The data for this model was from an early design study by McDonnell Douglas Astronautics Company for a dual-keel space station. The observer and its equations are developed in Chapter III. Chapter IV is the simulation runs of the observer versus the plant. Chapter V presents conclusions and recommendations for further research.

II. MATHEMATICAL MODEL

A. INTRODUCTION

Prior to the proposed space station almost all of the objects put into space could be treated as simple rigid bodies for the purpose of mathematical modelling of their motions. The design constraints imposed by the high cost of lifting mass to orbit dictates a light, open structure with considerable flexing. Large space structures such as the space station, therefore, cannot be treated as rigid bodies. The structure is in fact lightly damped with multiple natural frequencies. The result is a structure that will vibrate for considerable periods of time whenever external forces are applied.

The space station structure can be modeled as an n-DOF (degree of freedom) system consisting of n masses, springs, and dashpots [Ref. 2: p. 173-176]. This straight forward modelling of the coupled masses produces a system of unworkable complexity. As a result, the system will be modelled in terms of the structures natural modes of vibration. The resulting system, while still complex, is at least workable.

The model will be developed in two steps. The first will be to generate the continuous-time model of the natural modes. The second will yield the discrete-time model, developed from the first model, for use in the simulation.

B. MODAL MODEL

The space station structure can be modeled as a system of discrete masses coupled by springs and dashpots. The major mechanism of damping in the structure is structural damping, the internal dissipation of energy within the members, as the structure vibrates. Structural damping can be shown to be equivalent to viscous damping and this equivalency is used in the model [Ref. 2: p. 72-73].

The energy dissipated by structural damping is:

$$W_d = \alpha X^2 \quad (1)$$

W_d = energy dissipated by structural damping

α = constant (force/displacement)

X = displacement

The energy dissipated by viscous damping is:

$$W_v = \pi c \omega X^2 \quad (2)$$

We can equate the two

$$\pi C_{eq} \omega X^2 = \alpha X^2 \quad (3)$$

yielding an equivalent viscous damping coefficient:

$$C_{eq} = \frac{\alpha}{\pi \omega} \quad (4)$$

The second order differential equation for a single viscously damped mass is:

$$m\ddot{x} + c\dot{x} + kx = F(t) \quad (5)$$

Substituting C_{eq} for c

$$m\ddot{x} + \frac{\alpha}{\pi \omega} \dot{x} + kx = F(t) \quad (6)$$

For multiple mass systems C_{eq} becomes $\frac{d}{\omega_f} K$ where ω_f is the natural frequency of vibration.

The displacement of masses can be represented by the second order matrix differential equation [Ref. 3: p. 3-9],

$$M\ddot{q}(t) + \frac{d}{\omega_f} K\dot{q}(t) + Kq(t) = F(t) \quad (7)$$

q = coordinate vector

M = system mass matrix (diagonal)

$\frac{d}{\omega_f} K$ = equivalent damping

d = damping coefficient

ω_f = frequency of oscillation of the system

K = symmetric system stiffness matrix

$F(t)$ = system forcing function

The above equation represents a system of second order differential equations coupled through the stiffness matrix. Decoupling can be done by expressing q in terms of natural modes of vibration. The process is called modal analysis. The independent differential equations can then be treated individually. The modal equations are derived below.

First, the undamped, homogeneous form of Eq. (7)

$$\mathbf{M}\ddot{q}(t) + \mathbf{K}q(t) = 0 \quad (8)$$

is solved. Let

$$q(t) = Ax \sin(\omega t + \Theta) \quad (9)$$

$$\dot{q}(t) = Ax\omega \cos(\omega t + \Theta) \quad (10)$$

$$\ddot{q}(t) = -Ax\omega^2 \sin(\omega t + \Theta) \quad (11)$$

substituting Eq. (9) and Eq. (10) into Eq. (11)

$$[-\omega^2\mathbf{M} + \mathbf{K}]Ax \sin(\omega t + \Theta) = 0 \quad (12)$$

This equation has a non-trivial solution for all time if and only if:

$$[\mathbf{K} - \omega^2\mathbf{M}]x = 0 \quad (13)$$

Equation (12) has n combinations of x (natural mode shapes) and ω (natural frequencies) as solutions. These can be grouped into matrices:

$$\mathbf{X} = [x_1 x_2 \dots x_n]^T \quad (14)$$

$$\Omega^2 = \text{diag}[\omega_{o1}^2 \omega_{o2}^2 \dots \omega_{on}^2] \quad (15)$$

which satisfy the equation:

$$\mathbf{KX} = \Omega^2 \mathbf{MX} \quad (16)$$

Several useful relations can be derived from Eq. (16). Premultiplying Eq. (16) by \mathbf{X}^T ,

$$\mathbf{X}^T \mathbf{KX} = \Omega^2 \mathbf{X}^T \mathbf{MX} \quad (17)$$

The eigenvectors can be normalized

$$\mathbf{X}^T \mathbf{MX} = \mathbf{I} \quad (18)$$

which yields

$$\mathbf{X}^T \mathbf{KX} = \Omega^2 \quad (19)$$

The equations of motion can be uncoupled through the linear transformation of the coordinate system

$$q(t) = \sum_{i=1}^n x_i \eta_i(t) = \mathbf{X} \eta(t) \quad (20)$$

\mathbf{X} = modal matrix

n = maximum number of degrees of freedom

$\eta(t)$ = transformed coordinate vector

Application of the transformation to the system Eq. (7) yields

$$\mathbf{X}^T \mathbf{M} \mathbf{X} \ddot{\eta}(t) + \frac{d}{\omega_f} \mathbf{X}^T \mathbf{K} \mathbf{X} \dot{\eta}(t) + \mathbf{X}^T \mathbf{K} \mathbf{X} \eta(t) = \mathbf{X}^T \mathbf{F}(t) \quad (21)$$

Using Eq.(18) and Eq. (19)

$$\mathbf{X}^T \mathbf{M} \mathbf{X} \ddot{\eta}(t) = \mathbf{I} \ddot{\eta}(t) = \ddot{\eta} \quad (22)$$

$$\frac{d}{\omega_f} \mathbf{X}^T \mathbf{K} \mathbf{X} \dot{\eta}(t) = \frac{d}{\omega_f} \Omega^2 \dot{\eta}(t) = d\Omega \dot{\eta} \quad (23)$$

$$\mathbf{X}^T \mathbf{K} \mathbf{X} \eta(t) = \Omega^2 \eta \quad (24)$$

therefore

$$\ddot{\eta} + d\Omega \dot{\eta} + \Omega^2 \eta = \mathbf{X}^T \mathbf{F} \quad (25)$$

Equation (25) is the modal model of uncoupled second order differential equations. The motion of the structure can be found from the modal amplitudes, $\eta(t)$, using Eq. (20).

C. DISCRETE-TIME MODEL

The discrete-time state space model is found by solving the continuous-time equations. The i th equation of motion is

$$\ddot{\eta}_i(t) + d\omega_{oi} \dot{\eta}_i(t) + \omega_{oi}^2 \eta_i(t) = \mathbf{X}_i^T \mathbf{F}(t) \quad (26)$$

\mathbf{X}_i^T = transpose of the i th mode shape vector

$\mathbf{F}(t)$ = torquing force applied at a point

The homogeneous solution ($\mathbf{F}(t) = 0$) for Eq. (26) is [Ref. 4: p. 475-476]

$$\eta(t) = C_1 e^{-\gamma t} \cos(\omega_d t) + C_2 e^{-\gamma t} \sin(\omega_d t) \quad (27)$$

where

$$\gamma = \frac{d\omega_{oi}}{2} \quad (28)$$

$$\omega_d = \sqrt{\omega_{oi}^2 - \gamma^2} \quad (29)$$

The constants in Eq. (27) can be found by taking the derivative

$$\dot{\eta}(t) = (C_2 \omega_d - C_1 \gamma) e^{-\gamma t} \cos(\omega_d t) - (C_1 \omega_d - C_2 \gamma) e^{-\gamma t} \sin(\omega_d t) \quad (30)$$

and evaluating at $t = 0$

$$\eta(0) = C_1 \quad (31)$$

$$\dot{\eta}(0) = C_2 \omega_d - C_1 \gamma \quad (32)$$

Solving for C_1 and C_2

$$C_1 = \eta(0) \quad (33)$$

$$C_2 = \frac{\dot{\eta}(0)}{\omega_d} + \frac{\eta(0)\gamma}{\omega_d} \quad (34)$$

In matrix form

$$\begin{bmatrix} C_1 \\ C_2 \end{bmatrix} = \begin{bmatrix} 1 & 0 \\ \frac{\gamma}{\omega_d} & \frac{1}{\omega_d} \end{bmatrix} \begin{bmatrix} \eta(0) \\ \dot{\eta}(0) \end{bmatrix} \quad (35)$$

Rewriting Eq. (27) and Eq. (30) in matrix form

$$\begin{bmatrix} \eta(t) \\ \dot{\eta}(t) \end{bmatrix} = \begin{bmatrix} e^{-\gamma t} \cos(\omega_d t) & e^{-\gamma t} \sin(\omega_d t) \\ e^{-\gamma t} [\gamma \cos(\omega_d t) + \omega_d \sin(\omega_d t)] & e^{-\gamma t} [\omega_d \cos(\omega_d t) - \gamma \sin(\omega_d t)] \end{bmatrix} \begin{bmatrix} C_1 \\ C_2 \end{bmatrix} \quad (36)$$

Substituting Eq. (35) into Eq. (36), the solution can be written in terms of the initial conditions

$$\begin{bmatrix} \eta(t) \\ \dot{\eta}(t) \end{bmatrix} = \begin{bmatrix} e^{-\gamma t} [\cos(\omega_d t) + \frac{\gamma}{\omega_d} \sin(\omega_d t)] & \frac{1}{\omega_d} e^{-\gamma t} \sin(\omega_d t) \\ -\frac{\omega_o}{\omega_d} e^{-\gamma t} \sin(\omega_d t) & e^{-\gamma t} [\cos(\omega_d t) - \frac{\gamma}{\omega_d} \sin(\omega_d t)] \end{bmatrix} \begin{bmatrix} \eta(0) \\ \dot{\eta}(0) \end{bmatrix} \quad (37)$$

Letting

$$\mathbf{X}_i(t) = \begin{bmatrix} \eta_i(t) \\ \dot{\eta}_i(t) \end{bmatrix} \quad (38)$$

and

$$\mathbf{A}_i = \begin{bmatrix} e^{-\gamma t} [\cos(\omega_d t) + \frac{\gamma}{\omega_d} \sin(\omega_d t)] & \frac{1}{\omega_d} e^{-\gamma t} \sin(\omega_d t) \\ -\frac{\omega_o}{\omega_d} e^{-\gamma t} \sin(\omega_d t) & e^{-\gamma t} [\cos(\omega_d t) - \frac{\gamma}{\omega_d} \sin(\omega_d t)] \end{bmatrix} \quad (39)$$

the solution can be written as

$$\mathbf{X}_i(t) = \mathbf{A}_i(t) \mathbf{X}_i(0) \quad (40)$$

where \mathbf{A}_i is the state transition matrix of the i th mode. The non-homogeneous solution is

$$\mathbf{X}_i(t) = \mathbf{A}_i(t) \mathbf{X}_i(0) + \mathbf{B}_i x_i^T \mathbf{F}(0) \quad (41)$$

where the discrete-time input matrix, for constant \mathbf{F} , is given by

$$\mathbf{B}_i = \int_0^T \mathbf{B}_i(\tau) \Gamma \partial \tau \quad (42)$$

and $\Gamma = [0 \ 1]^T$ is the input matrix for the continuous-time system, and T is the sampling time. Solving Eq. (42) yields

$$\mathbf{B}_i = \begin{bmatrix} \frac{1}{\omega_o^2} [1 - e^{-\gamma T} \cos(\omega_d T) - \frac{\gamma}{\omega_d} e^{-\gamma T} \sin(\omega_d T)] \\ \frac{1}{\omega_d} e^{-\gamma T} \sin(\omega_d T) \end{bmatrix} \quad (43)$$

The discrete-time state equation for the i th equation of motion can be written as

$$\mathbf{X}_i(kT + 1) = \mathbf{A}_i(T) \mathbf{X}_i(kT) + \mathbf{B}_i(T) x_i^T \mathbf{F}(kT) \quad (44)$$

where \mathbf{A}_i and \mathbf{B}_i are evaluated at $t = T$. Here,

\mathbf{X}_i = vector of the i th modal amplitude and the i th modal velocity

\mathbf{A}_i = ith state transition matrix

\mathbf{B}_i = ith input vector

x_i^T = transpose of the ith mode shape vector

\mathbf{F} = distributed force on the plant

T = sampling time

k = time index

Equation (44) can be expanded to include the disturbance input, $w(kT)$:

$$\mathbf{X}_i(kT + 1) = \mathbf{A}_i(T)\mathbf{X}_i(kT) + \mathbf{B}_i(T)x_i^T[\mathbf{F}(kT) + w(kT)] \quad (45)$$

Equation (45) is the discrete-time mathematical model describing the motion of the structure in terms of its natural modes of vibration.

III. THE OBSERVER

A. INTRODUCTION

The observer design will be required to estimate the modal states in a noisy environment. Kalman filtering is the most widely used technique for accomplishing the production of state estimates in a noisy environment [Ref. 5: p.159]. The steady state Kalman filter was selected to minimize the computations during the actual plant observer operation. Use of a steady-state gain matrix for the observer allows the matrix to be computed separately from the operational observer, reducing the computer power required for the observer and allowing the algorithm to operate more rapidly.

B. KALMAN FILTER EQUATIONS

The discrete Kalman filter provides state estimates for the following dynamic system [Ref. 5: p. 159-162],

$$\mathbf{X}(k+1) = \mathbf{AX}(k) + \mathbf{BU}(k) + \mathbf{BnW}(k) \quad (46)$$

$$\mathbf{Y}(k+1) = \mathbf{CX}(k+1) + \mathbf{V}(k+1) \quad (47)$$

\mathbf{X} = $n \times 1$ state vector

\mathbf{U} = $P \times 1$ control vector

\mathbf{W} = $r \times 1$ plant noise vector

\mathbf{Y} = $m \times 1$ measurement vector

\mathbf{V} = $m \times 1$ measurement noise vector

\mathbf{A} = $n \times n$ state transition matrix

\mathbf{B} = $n \times p$ control input matrix

\mathbf{Bn} = $n \times r$ plant noise input matrix

\mathbf{C} = $m \times n$ measurement matrix

The plant noise vector $\mathbf{W}(k)$ is gaussian white noise with

$$\mathbf{E}\{\mathbf{W}(k)\} = \mathbf{0} \quad (48)$$

$$\mathbf{E}\{\mathbf{W}(k)\mathbf{W}^T(k)\} = \mathbf{Q} \quad (49)$$

for all $k = 0, 1, 2, \dots$, and \mathbf{Q} is a positive semi-definite $r \times r$ matrix. $\mathbf{V}(k)$ is gaussian white noise with

$$\mathbf{E}\{\mathbf{V}(k)\} = \mathbf{0} \quad (50)$$

$$\mathbf{E}\{\mathbf{V}(k)\mathbf{V}^T(k)\} = \mathbf{R} \quad (51)$$

for all $k = 0, 1, 2, \dots$, and \mathbf{R} is a positive definite $m \times m$ matrix. The two random processes $\mathbf{W}(k)$ and $\mathbf{V}(k)$ are assumed to be independent, so that

$$\mathbf{E}\{\mathbf{V}(j)\mathbf{W}(k)\} = \mathbf{0} \quad (52)$$

for all $j = 1, 2, \dots$, and $k = 0, 1, 2, \dots$. The intial state $\mathbf{X}(0)$ is assumed to be a gaussian random vector with

$$\mathbf{E}\{\mathbf{X}(0)\} = \mathbf{0} \quad (53)$$

It is assumed that $\mathbf{X}(0)$ is independent of $\mathbf{W}(k)$ and $\mathbf{V}(k)$.

The optimal estimate of $\mathbf{X}(k + 1)$ is denoted $\hat{\mathbf{X}}(k + 1 | k + 1)$. The Kalman filter is designed to minimize

$$\mathbf{J} = \mathbf{E}\{[\mathbf{X}(k + 1) - \hat{\mathbf{X}}(k + 1 | k + 1)]^T[\mathbf{X}(k + 1) - \hat{\mathbf{X}}(k + 1 | k + 1)]\} \quad (54)$$

The recursive realtions for generating $\hat{\mathbf{X}}(k + 1 | k + 1)$ are

$$\hat{\mathbf{X}}(k + 1 | k) = \mathbf{A}\hat{\mathbf{X}}(k | k) + \mathbf{B}\mathbf{U}(k) \quad (55)$$

$$\hat{\mathbf{X}}(k + 1 | k + 1) = \hat{\mathbf{X}}(k + 1 | k) + \mathbf{G}(k + 1)[\mathbf{Y}(k + 1) - \mathbf{C}\hat{\mathbf{X}}(k + 1 | k)] \quad (56)$$

for $k = 0, 1, 2, \dots$, where $\hat{\mathbf{X}}(0 | 0) = \mathbf{0}$. $\hat{\mathbf{X}}(0 | 0)$ is set equal to zero since the expectation of $\mathbf{X}(0)$ is zero.

$\mathbf{G}(k + 1)$ is an $n \times m$ matrix, called the Kalman Gain Matrix which is specified by the realtions:

$$\mathbf{P}(k + 1 | k) = \mathbf{A}\mathbf{P}(k | k)\mathbf{A}^T + \mathbf{B}\mathbf{Q}(k)\mathbf{B}^T \quad (57)$$

$$\mathbf{G}(k + 1) = \mathbf{P}(k + 1 | k)\mathbf{C}^T[\mathbf{C}\mathbf{P}(k + 1 | k)\mathbf{C}^T + \mathbf{R}(k + 1)]^{-1} \quad (58)$$

$$\mathbf{P}(k + 1 | k + 1) = [\mathbf{I} - \mathbf{G}(k + 1)\mathbf{C}]\mathbf{P}(k + 1 | k) \quad (59)$$

$P(k | k)$ is the covariance matrix of the error between the states and their estimates

$$P(k | k) = E\{[X(k) - \hat{X}(k | k)][X(k) - \hat{X}(k | k)]^T\} \quad (60)$$

Since we are using the steady state gains the choice of $P(0 | 0)$ is irrelevant. $P(0 | 0)$ is initialized to zero in the gain derivation program for simplicity [Ref. 6: p. 139-140].

C. STEADY-STATE SOLUTION

If Equations (57), (58), and (59) are repeatedly iterated, $G(k + 1)$ will converge to a steady state value [Ref. 7: p. 263].

$$G_{ss} = \lim_{k \rightarrow \infty} G(k + 1) \quad (61)$$

The values of G_{ss} (or G) can be substituted into Eq. (56) making the steady state Kalman filter

$$\hat{X}(k + 1 | k) = A\hat{X}(k | k) + BU(k) \quad (62)$$

$$\hat{X}(k + 1 | k + 1) = \hat{X}(k + 1 | k) + G[Y(k + 1) - C\hat{X}(k + 1 | k)] \quad (63)$$

D. OBSERVER PERFORMANCE

The performance of an observer is judged by how accurately and rapidly it estimates the desired states. The performance measure of the observer as a whole is shown in equation (54). The normalized performance of the observer for individual states is

$$J_i = E[(x_i - \hat{x}_i)^2] / E[x_i^2] \quad (64)$$

which can be found using Eq. (65)

$$J_i = \sum_{k=0}^{\infty} (x_i(k) - \hat{x}_i(k))^2 T_s \div \sum_{k=0}^{\infty} x_i^2(k) T_s \quad (65)$$

J_i = performance measure for the i th state

$x_i(k)$ = value of the i th state at k

$\hat{x}_i(k)$ = observer estimate of i th state at k

T_s = sample interval

A normalized performance measure is used to aid comparison of the performance of the observer in estimating various states. From Eq. (65) it can be shown that if $\hat{x}_i(k) = 0$ for all $k = 0, 1, 2, \dots$ that J_i would be unity. Therefore, the better the performance of the observer, the smaller the fraction of one J_i will be.

IV. SIMULATION

A. INTRODUCTION

The objectives of the simulation were to

- determine the sensitivity of the observer performance and settling time to changes in the ratio of plant noise to measurement noise,
- determine the effect on observer performance and settling time of increasing the number of modes observed in the matched plant/observer, and
- determine the performance for the reduced order observer.

B. PLANT AND OBSERVER DATA

The dynamic model is a truncated form of a preliminary space station configuration; the phase II dual keel structure.¹ The full model consists of an infinite number of natural modes but this was restricted to the first ten active modes for this study due to limitations on computer resources. As will be shown reasonable data can be obtained with this simplification in examining the observer performance.

C. SIMULATION PROGRAMS

The simulation was broken down into two segments due to the large memory and computational time requirements. The first program computed the steady state observer gain matrix (G). The second program ran the observer and the plant when the plant was subjected to an impulse excitation.

The steady state observer gain matrix (G) was obtained by repeated iteration of equations (57), (58), and (59). The equations were run until the values of the matrix changed by less than a set fraction. The following formula was used to check the changes in the gain matrix elements

$$\Delta g_{i,j} = [g_{i,j}(k+1) - g_{i,j}(k)]/g_{i,j}(k+1) \quad (66)$$

The program was terminated when $\delta g_{i,j}$ was less than 10^{-10} .

The settling time for the estimates of the states to be within 2% of the actual states was determined by finding the eigenvalues of $A - G^*C$ then computing as follows [Ref. 6: p. 139-143]

¹ The model for preliminary station configuration was provided courtesy of McDonnell Douglas Astronautics Company, 5301 Bolsa Avenue, Huntington Beach, CA 92647.

$$T_s = \log(.02) / \log(\lambda_{AGC_{\min}}) \quad (67)$$

The expected error in the sensor, i.e., the standard deviation of the noise in the measurement, was chosen as 10^{-3} feet per sec. per sec. based on the natural frequencies in the structure and reasonable sensor sensitivity [Ref. 2: p. 79-80]. The expected plant noise was varied to find the range of ratios between plant and measurement noise that the filter would be effective. This approach was taken since the plant noise contributors are not currently well defined.

The second program subjected the plant as modeled in Eq. (45) to an impulse excitation and then had the observer estimate the selected states using observer equations (62) and (63). Observer performance was computed using Eq. (65).

A third program was used to find the contribution of unobserved modes to the noise in the Kalman observer. The program ran the plant subject to an impulse excitation and computed the product of the measurement matrix \mathbf{C} times the unobserved modes of the state vector $\mathbf{X}(k)$ for a measure of the noise contributed by the unobserved modes.

The three programs are listed in the appendices.

D. EFFECT OF PLANT TO MEASUREMENT NOISE RATIO ON OBSERVER PERFORMANCE

The ratio of the variance of the plant noise (PN) to the variance of the measurement noise (MN) was found to have a strong effect on the Kalman Observer performance (J) and settling time (T_s). Figures 1 through 6 show the observer performance for a 3 mode matched plant and observer system for progressively smaller PN/MN ratios.² Figure 7 shows the performance for the seventh mode (position) versus several values of PN/MN. Figure 8 is the settling time versus the same PN/MN ratios.

The figures show that, for all of the plotted performance values, the observer performance is at least marginally acceptable regardless of the PN/MN ratio. Decreasing the PN/MN ratio leads to an even more rapid degradation in observer performance. The settling times also rapidly increase as the PN/MN ratio decreases.

² Figures 1-6 and 9-15 show the performance measure for each mode. The bar for the mode position is immediately to the right of the numbered tick mark on the x-axis scale, the mode velocity is next to it immediately adjacent to the tick mark without a number.

E. EFFECTS OF INCREASED MODES ON OBSERVER PERFORMANCE

The matched plant/observer was run with increasing numbers of modes to see if there was an effect on observer performance (J) or settling time T_s . Figures 7 through 17 are of observer performance for systems with increasing numbers of modes in the system being observed. Figure 18 is of settling time versus the number of modes in the system. The ratio of PN/MN was kept constant at $PN/MN = 2.5 \times 10^9$.

The increasing of the number of modes for the matched plant/observer had negligible effect on the performance for the individual modes. The performance value for the modes was effectively constant. Settling times for the observers increased as the number of modes was increased.

F. REDUCED ORDER KALMAN OBSERVER

The Kalman Observer has been shown to be effective where the number of modes observed matches the number of modes in the plant. The Kalman Observer was then run with the one less mode observed than the number of modes in the plant. The gain matrix (G) from the matched system was used. The observer failed with the state estimates produced by the observer becoming excessively large and having settling times of hours vice minutes. Since the purpose of the observer was to provide estimates for use in controlling the plant the time delay makes the estimates unusable.

The cause of the observer failure is apparent when you look at the last portion of Eq. (56) of the Kalman Observer

$$G[Y(k+1) - \hat{C}\hat{X}(k+1|k)] \quad (68)$$

This portion of the observer equation is the correction of $\hat{X}(k+1|k)$ to produce $\hat{X}(k+1|k+1)$. The design of the Kalman observer is to produce an estimate despite the measurement noise but, with the reduced order filter there is additional unanticipated noise which causes over correction of the values of \hat{X} leading to the state estimates being excessively large and settling times being too long. This can be shown by examining what composes $Y(k+1) - CX(k+1|k)$

$$\begin{bmatrix}
 x_1(k) \\
 x_2(k) \\
 \uparrow \\
 \downarrow \\
 x_{m-1}(k) \\
 x_m(k) \\
 \cdots \\
 x_{m+1}(k) \\
 x_{m+2}(k) \\
 \uparrow \\
 \downarrow \\
 x_{n-1}(k) \\
 x_n(k)
 \end{bmatrix} - C \begin{bmatrix}
 \hat{x}_1(k) \\
 \hat{x}_2(k) \\
 \uparrow \\
 \downarrow \\
 \hat{x}_{m-1}(k) \\
 x_m(k) \\
 \cdots \\
 0 \\
 \uparrow \\
 \downarrow \\
 0
 \end{bmatrix} \quad (69)$$

C times the state $x_{m-1}(k)$ through $x_n(k)$ is unanticipated noise so if

$$C \begin{bmatrix}
 x_1(k) \\
 x_2(k) \\
 \uparrow \\
 \downarrow \\
 x_{m-1}(k) \\
 x_m(k)
 \end{bmatrix} - C \begin{bmatrix}
 \hat{x}_1(k) \\
 \hat{x}_2(k) \\
 \uparrow \\
 \downarrow \\
 \hat{x}_{m-1}(k) \\
 \hat{x}_m(k)
 \end{bmatrix} = 0 \quad (70)$$

the remaining portion of the C matrix times the modal states is an equivalent noise.

Table (1) shows the growth of the unanticipated noise in the filter as the number of unobserved modes in the plant grows. Table (2) shows the individual contributions of the individual modes when left unobserved. Table (1) shows that the unanticipated noise is much larger than that expected by the filter (10^{-3}). Table (2) shows that there are modes that do not markedly contribute to the noise and that they might successfully be left unobserved if the measurement noise estimate was already much larger than these noise sources.

Table 1. CUMULATIVE UNANTICIPATED NOISE FROM UNOBSERVED MODES

Number of Unobserved Modes	Unobserved Modes	E1	E2	E2
1	10	0.647	97.440	3.277
2	10-11	366.354	95.764	3.142
3	10-12	366.355	95.764	3.142
4	10-13	365.565	95.855	3.143
5	10-14	365.426	96.032	5.704
6	10-15	144195.7	116.205	2201.94
7	10-16	148006.8	170.142	5475.21
8	10-17	473974.3	39424.1	5692.50
9	10-18	474344.2	60078.8	9419.77
10	10-19	474358.5	68987.7	9865.27

Table 2. UNANTICIPATED NOISE FROM UNOBSERVED MODES BY MODE

Unobserved Mode	E1	E2	E3
10	0.64716	97.4403	3.27761
11	359.342	0.20929	0.21429
12	0.9353E-08	0.1364E-07	0.2196E-07
13	0.3852E-02	0.4409E-02	0.9017E-03
14	0.5615E-03	0.2592E-01	2.57554
15	143090.9	17.9682	2167.02
16	195.736	83.3458	5675.91
17	324471.2	39252.26	221.170
18	216.240	21133.6	3736.93
19	7.69298	8829.95	458.504
20	2.3194	3949.16	108.981

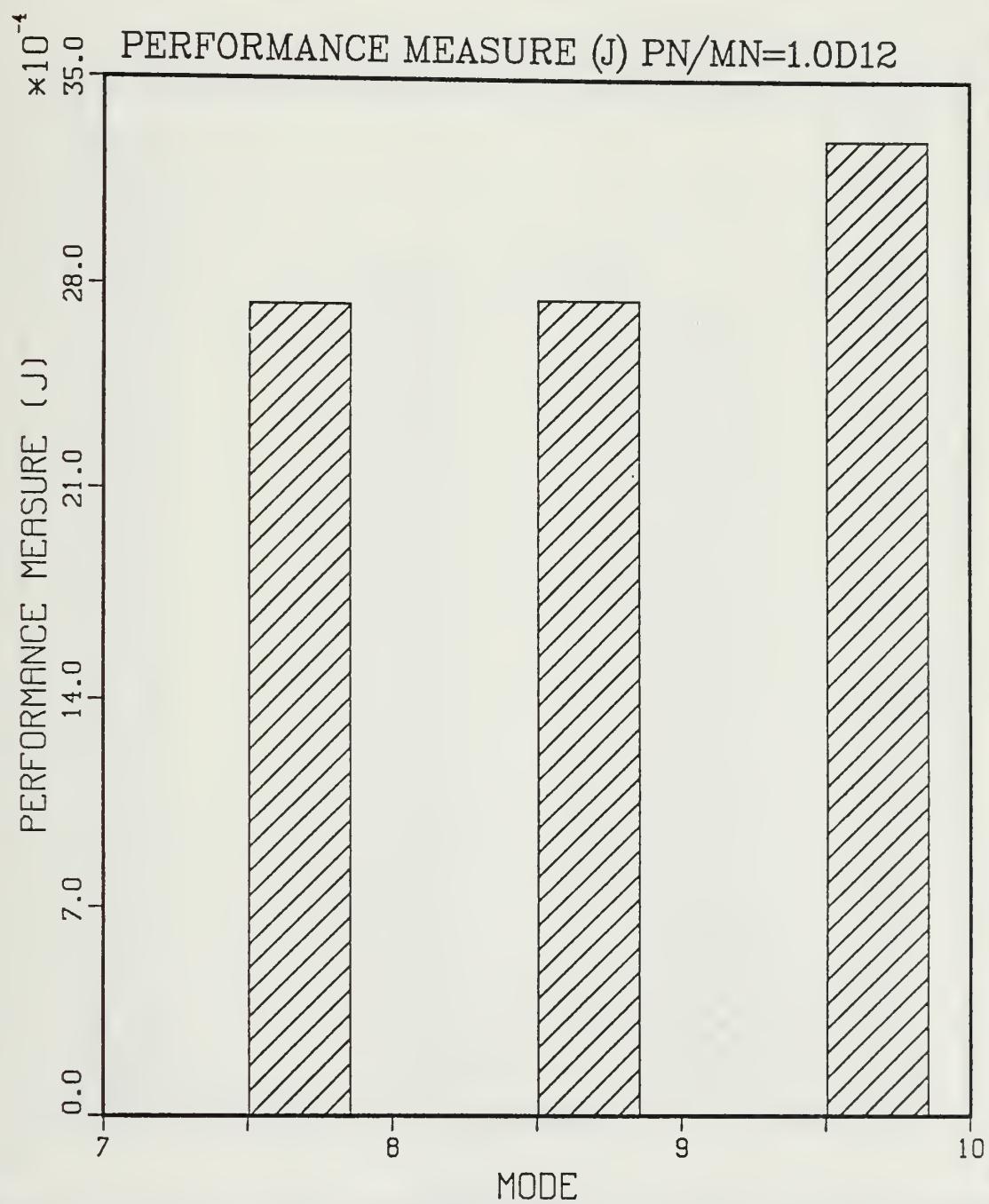


Figure 1. Observer Performance (J) $PN/MN = 1.0d12$

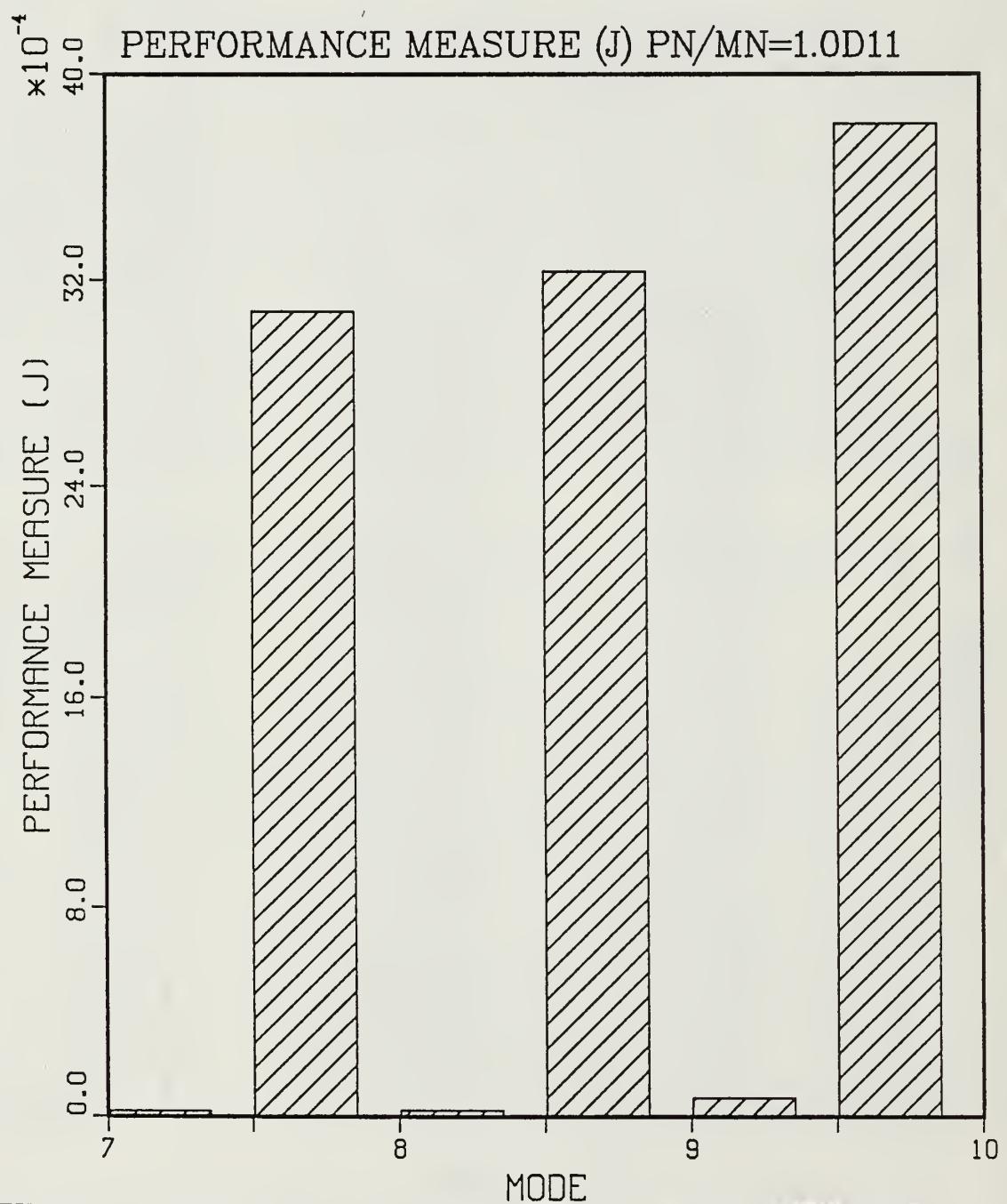


Figure 2. Observer Performance (J) PN/MN = 1.0d11

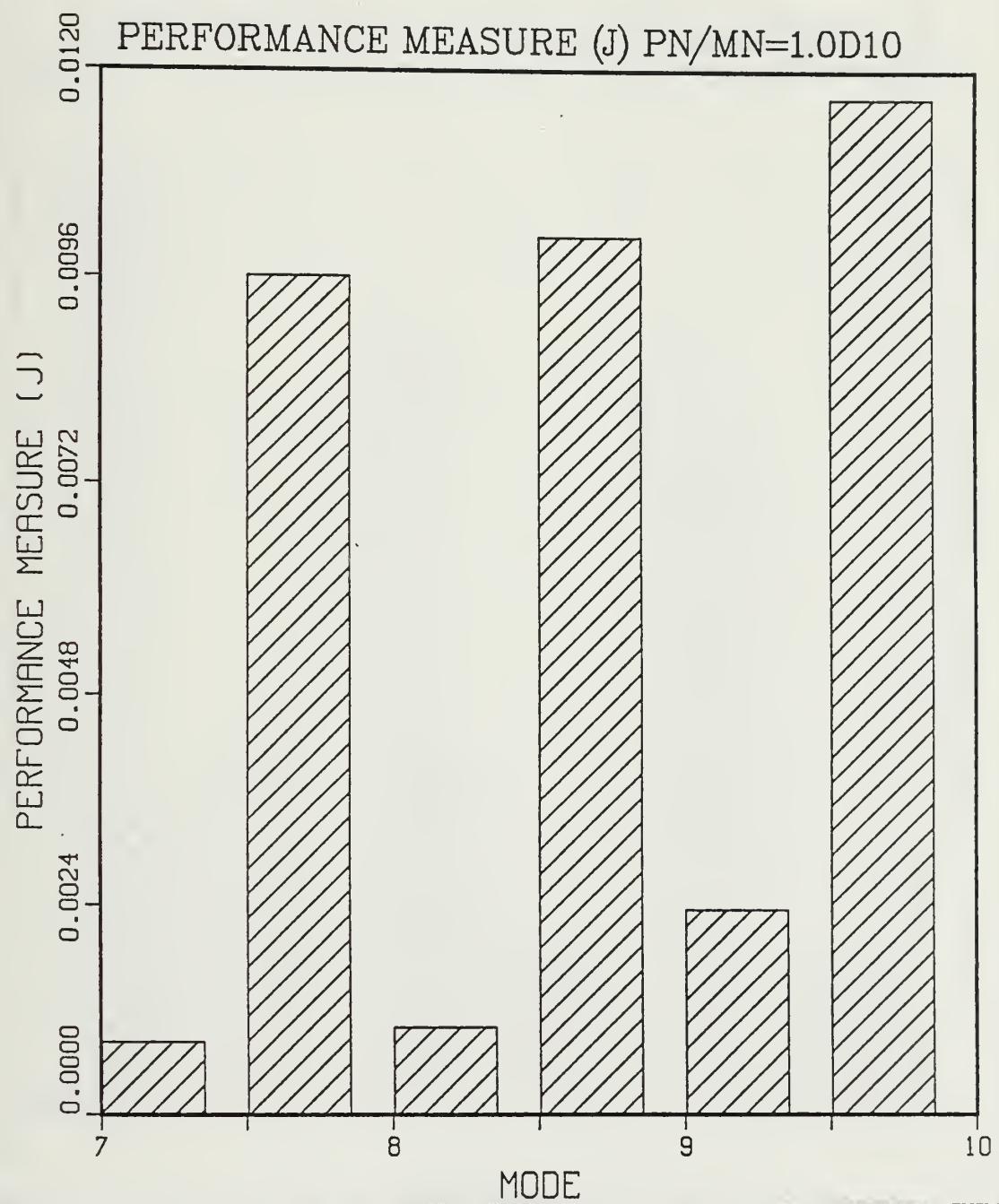


Figure 3. Observer Performance (J) PN/MN = 1.0d10

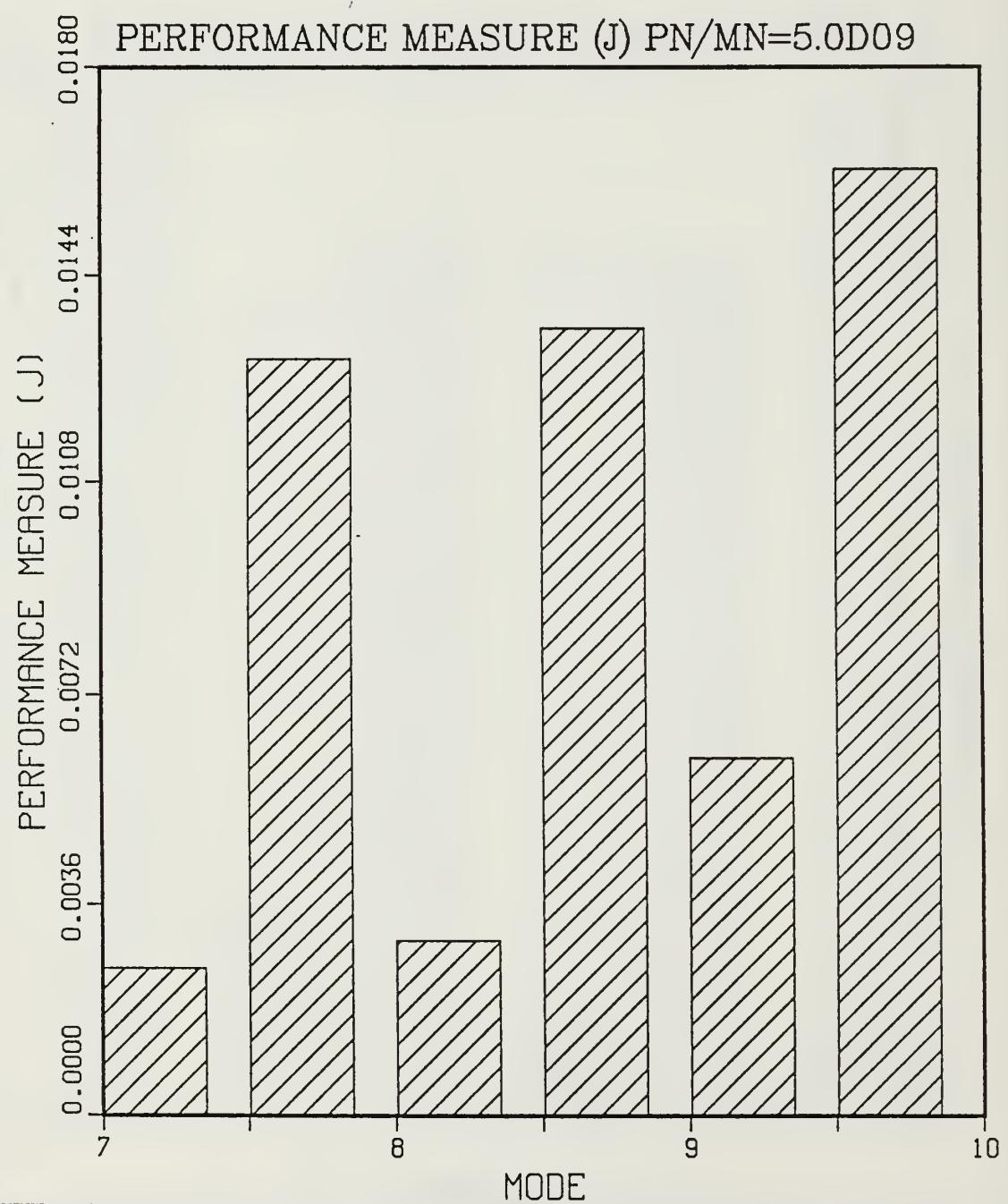


Figure 4. Observer Performance (J) PN/MN = 5.0d09

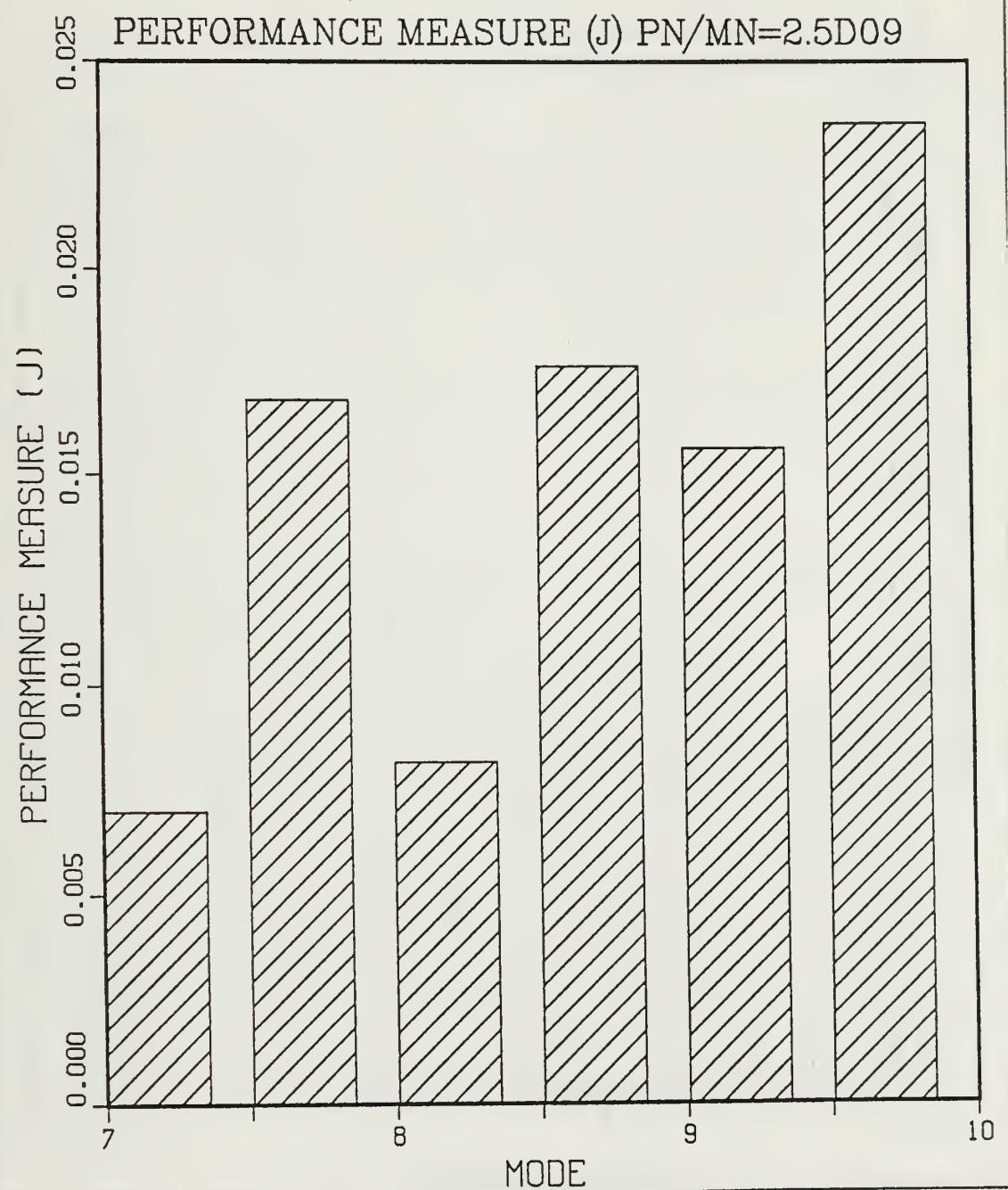


Figure 5. Observer Performance (J) PN/MN = 2.5d09

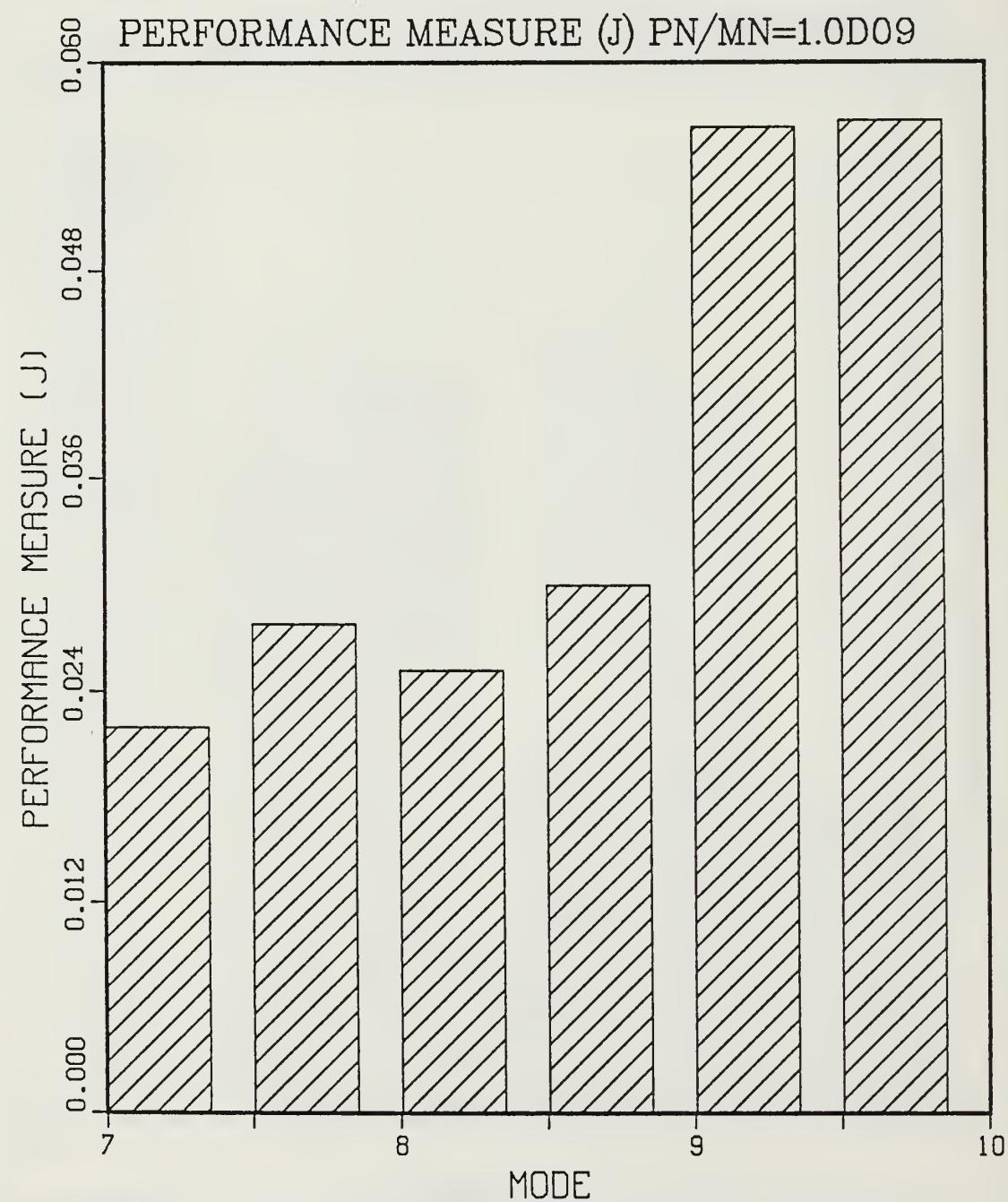


Figure 6. Observer Performance (J) PN/MN = 1.0d09

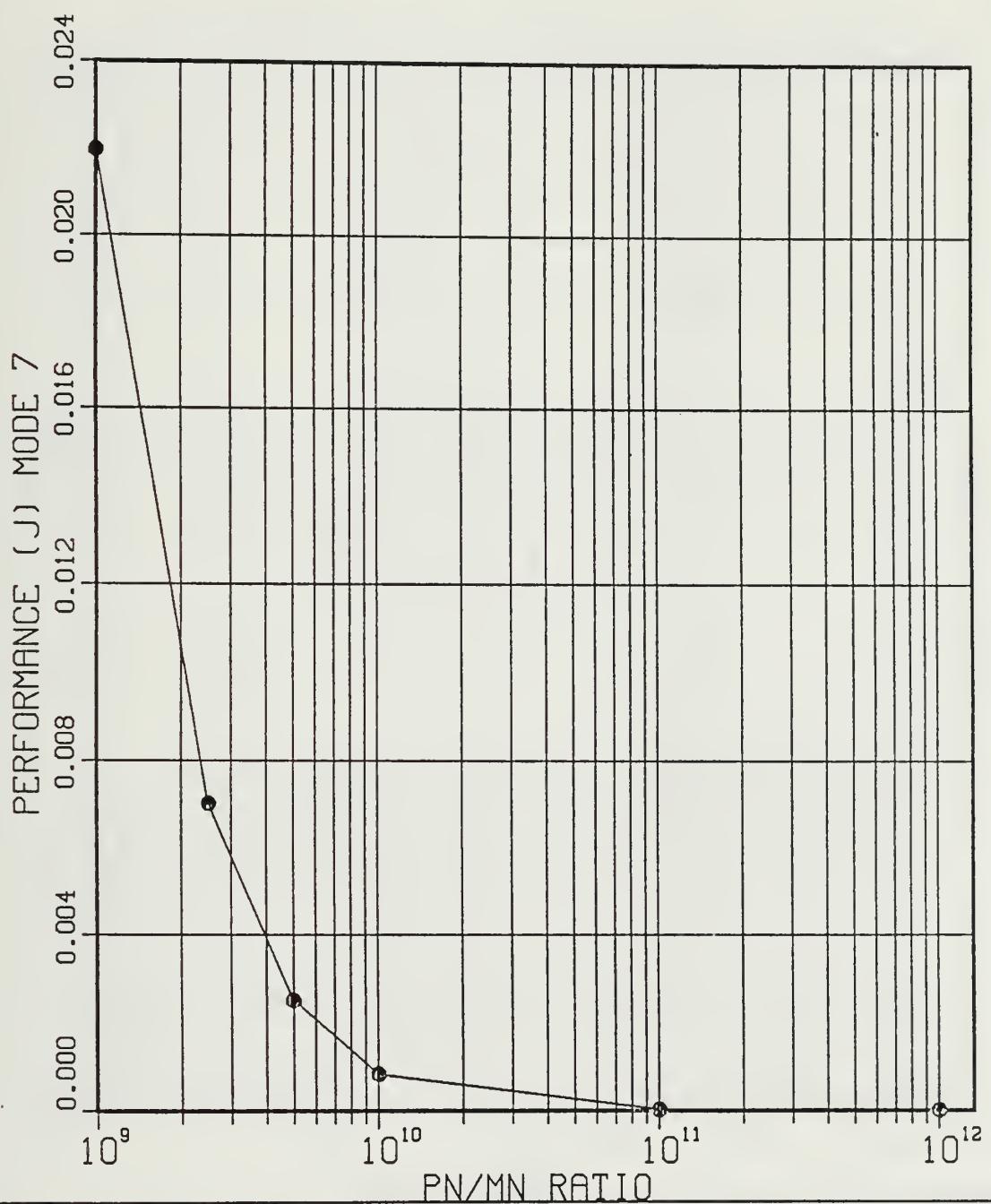


Figure 7. Mode 7 (Postion) Observer Performance versus PN/MN

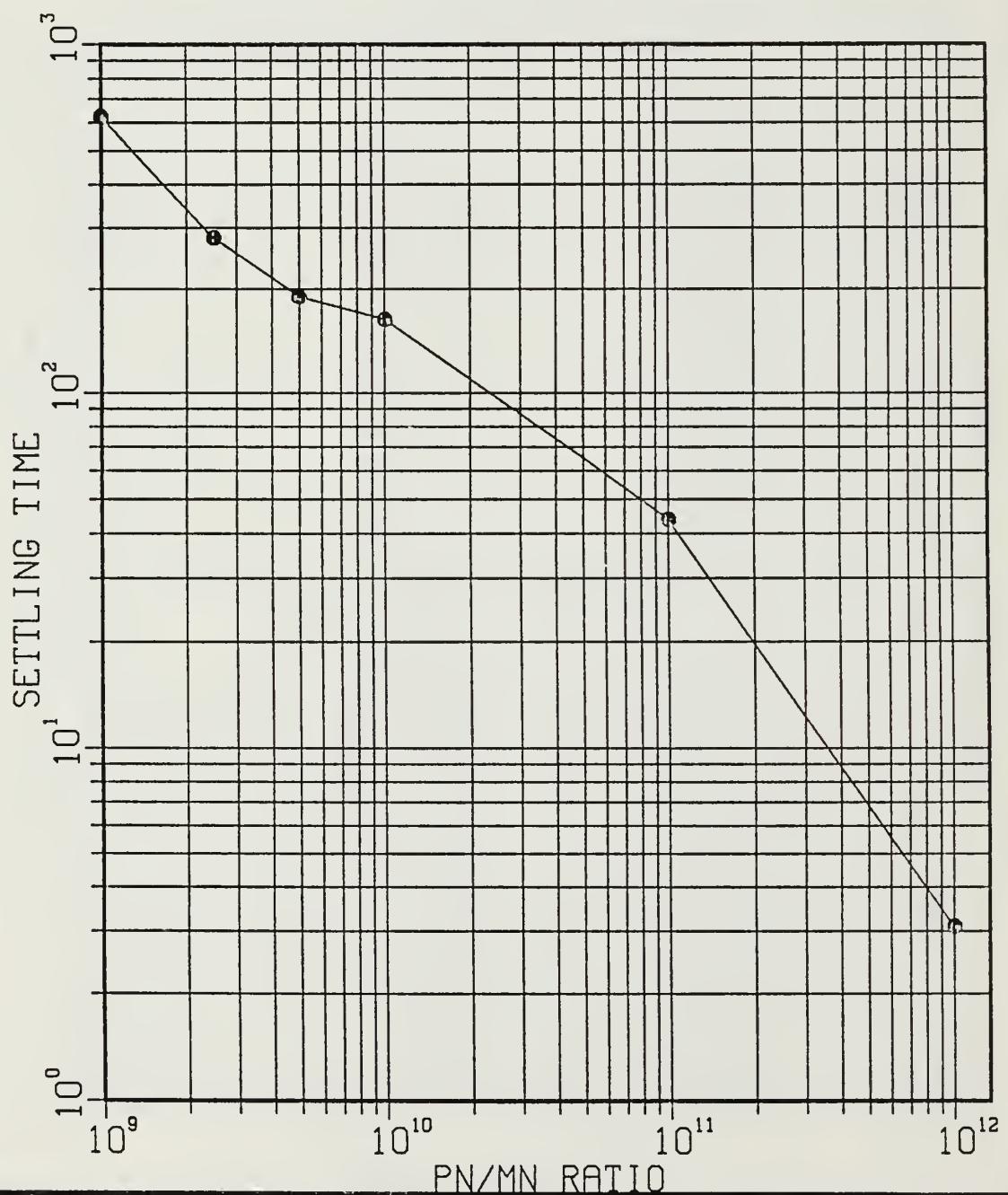


Figure 8. Settling Time versus PN/MN

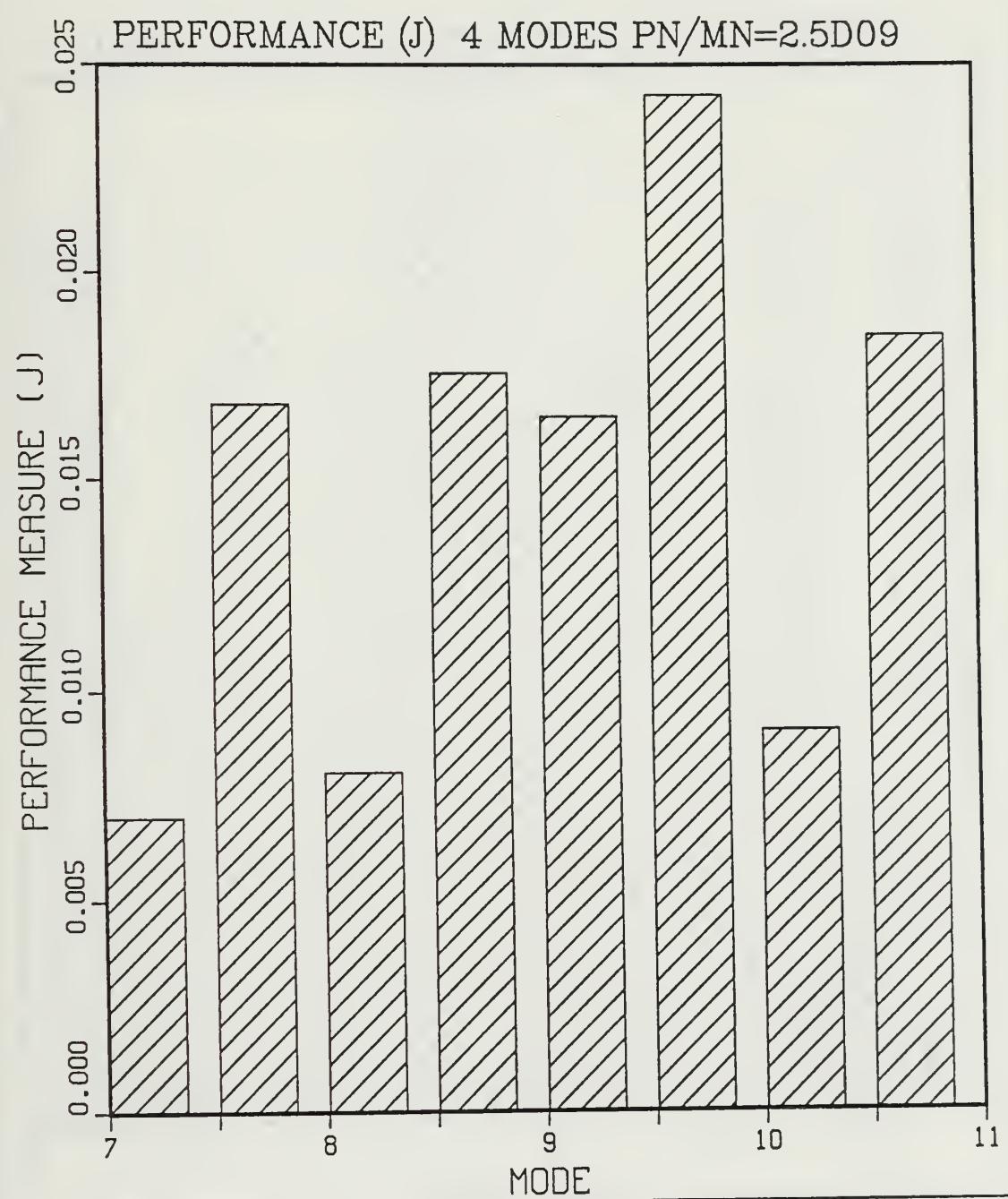


Figure 9. Observer Performance (J) 4 Modes (7 - 10)

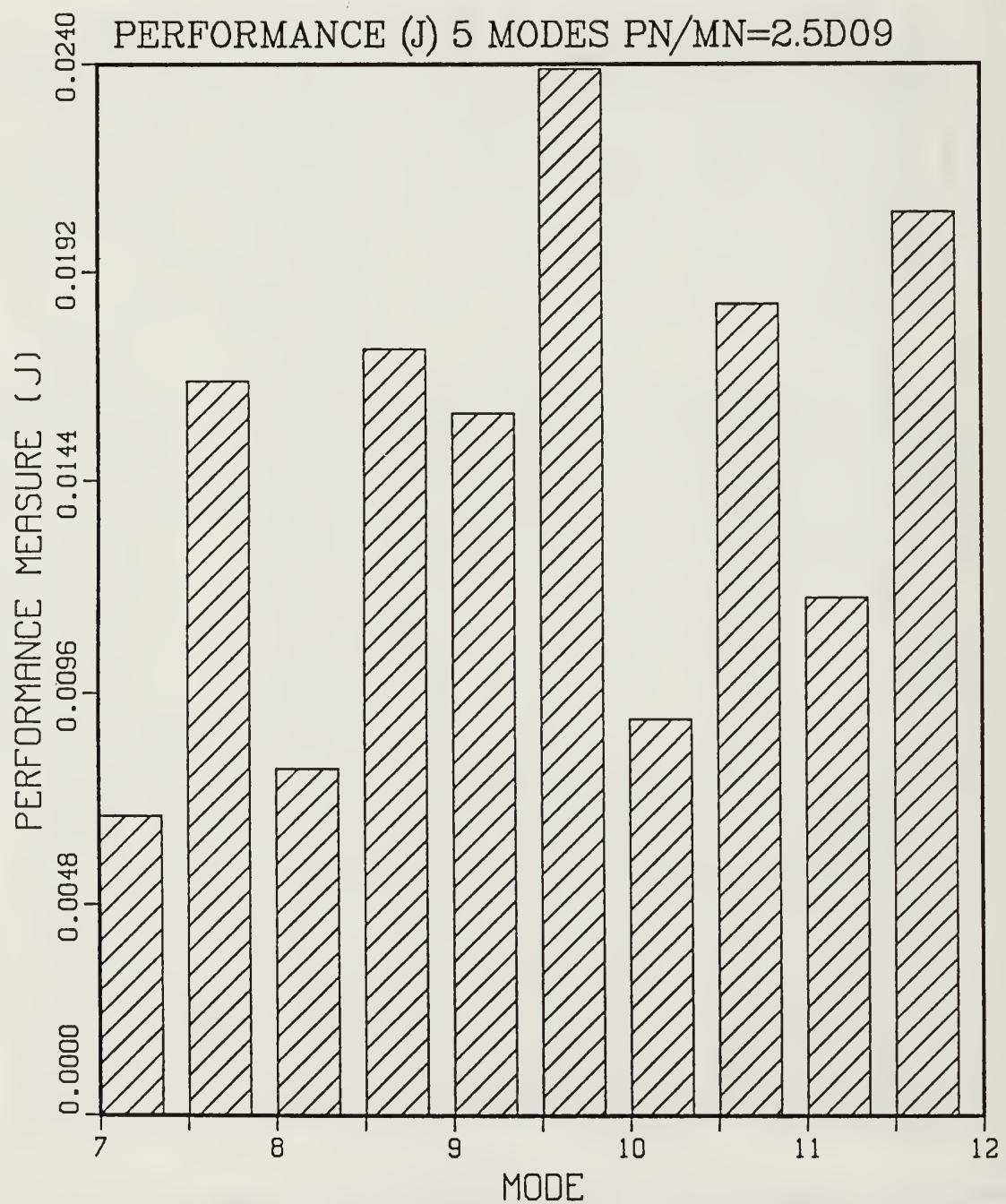


Figure 10. Observer Performance (J) 5 Modes (7 - 11)

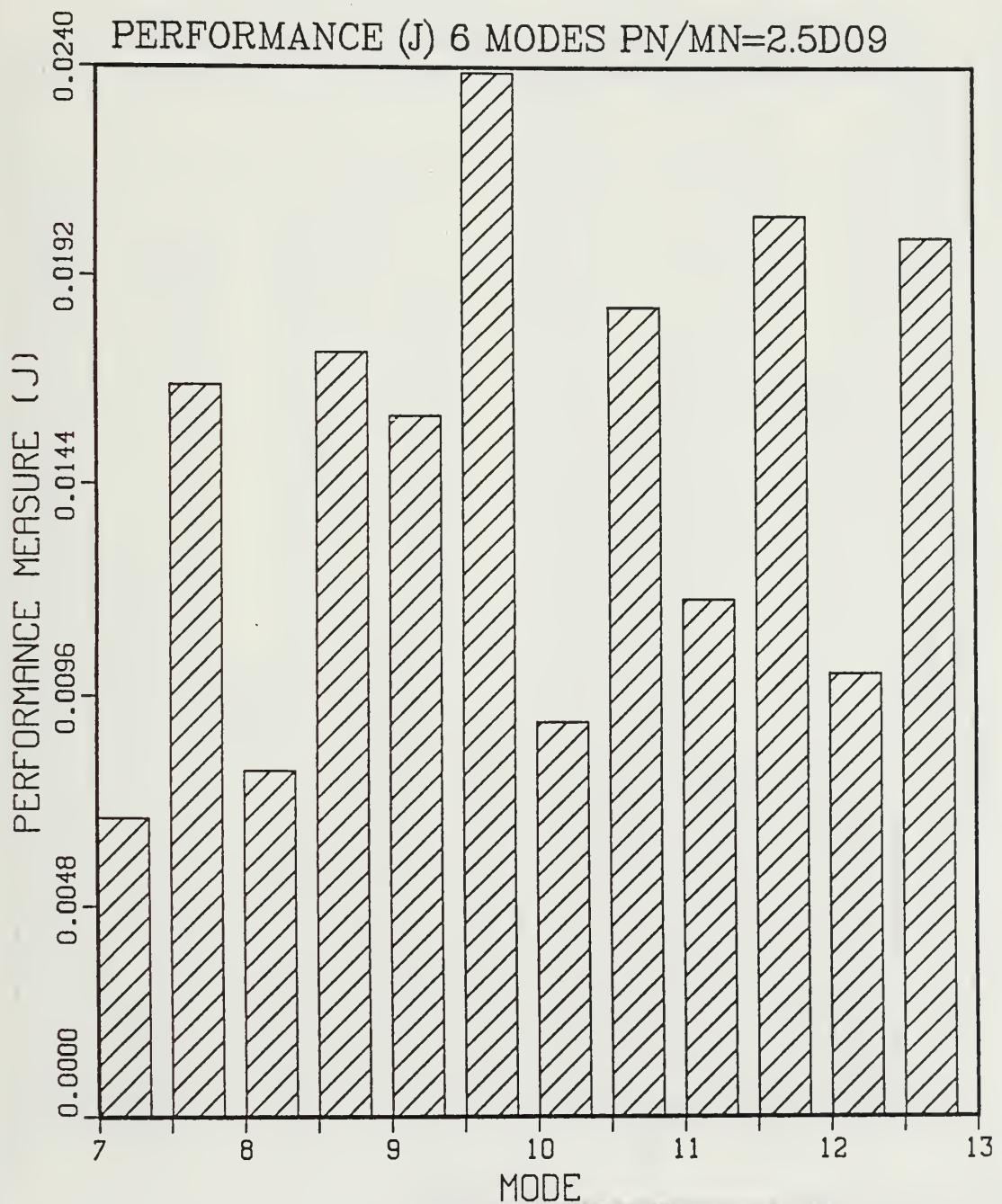


Figure 11. Observer Performance (J) 6 Modes (7 - 12)

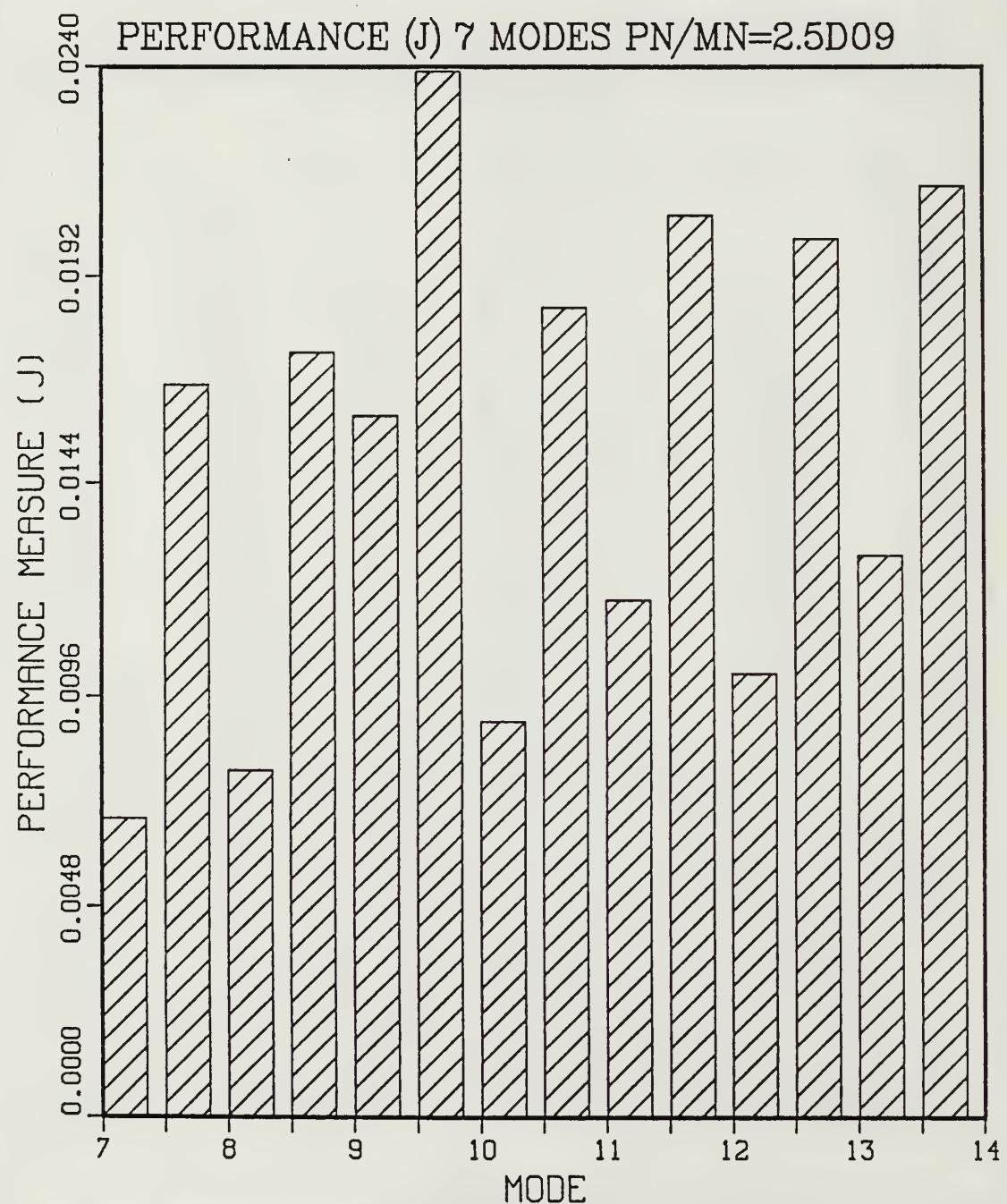


Figure 12. Observer Performance (J) 7 Modes (7 - 13)

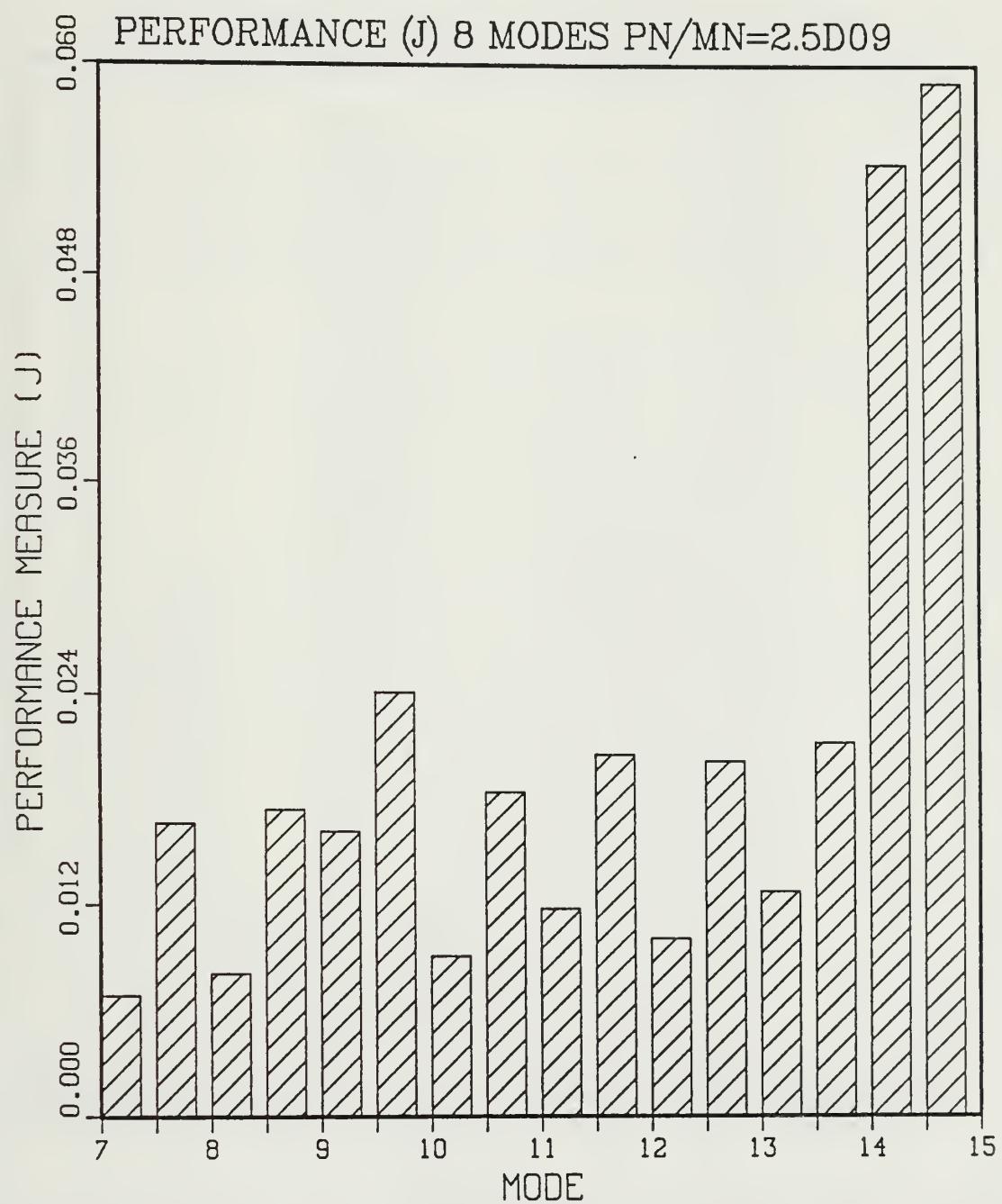


Figure 13. Observer Performance (J) 8 Modes (7 - 14)

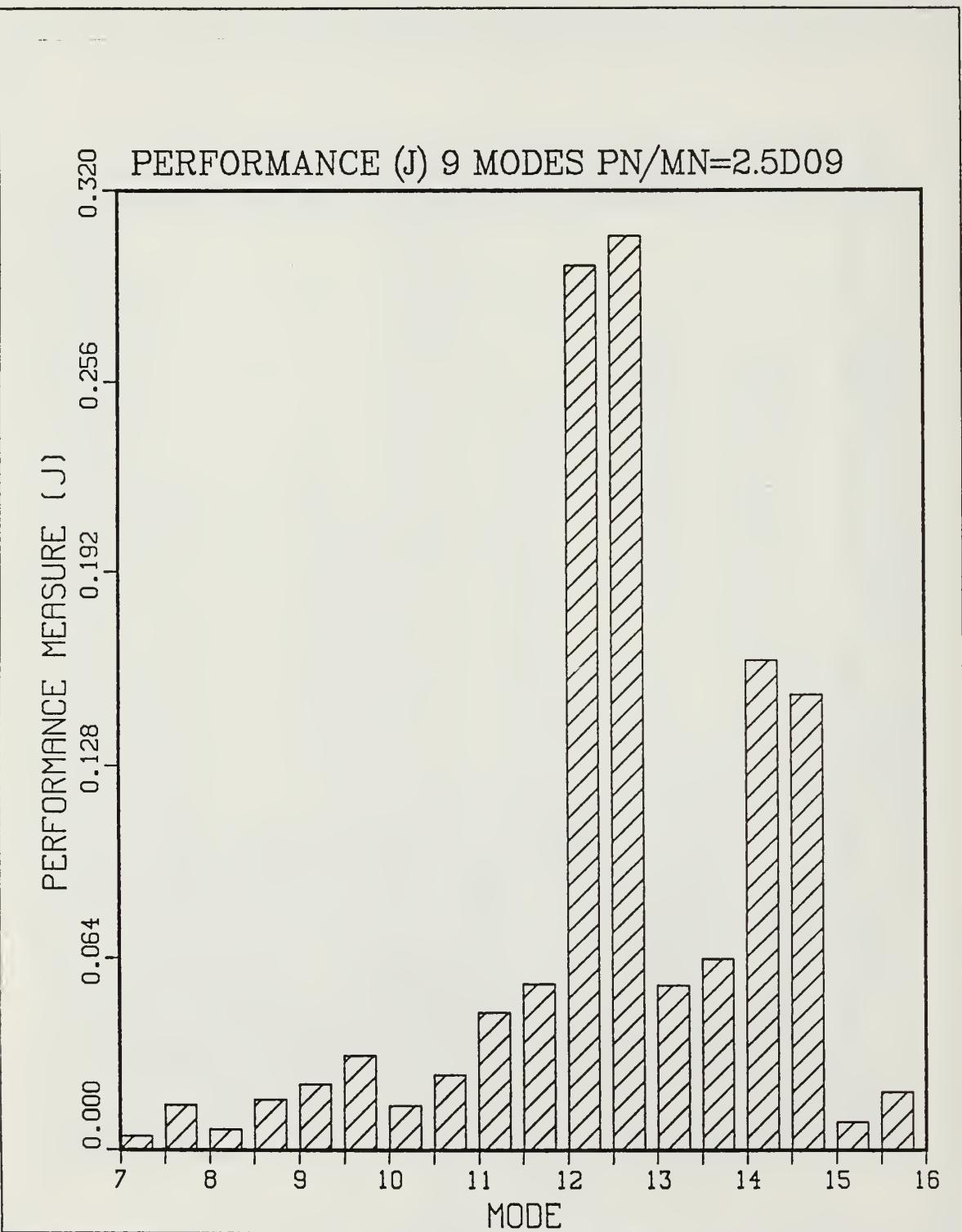


Figure 14. Observer Performance (J) 9 Modes (7 - 15)

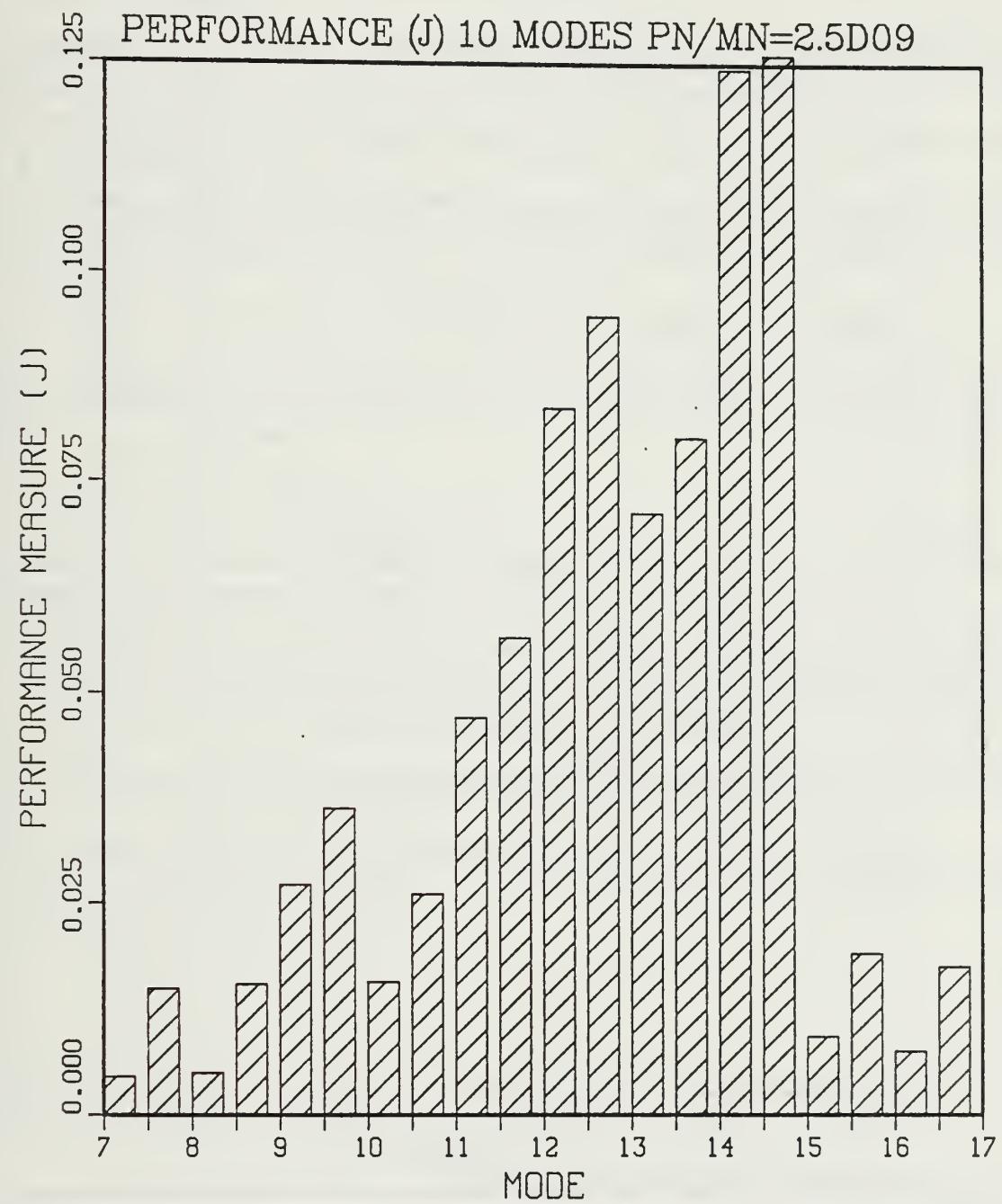


Figure 15. Observer Performance (J) 10 Modes (7 - 16)

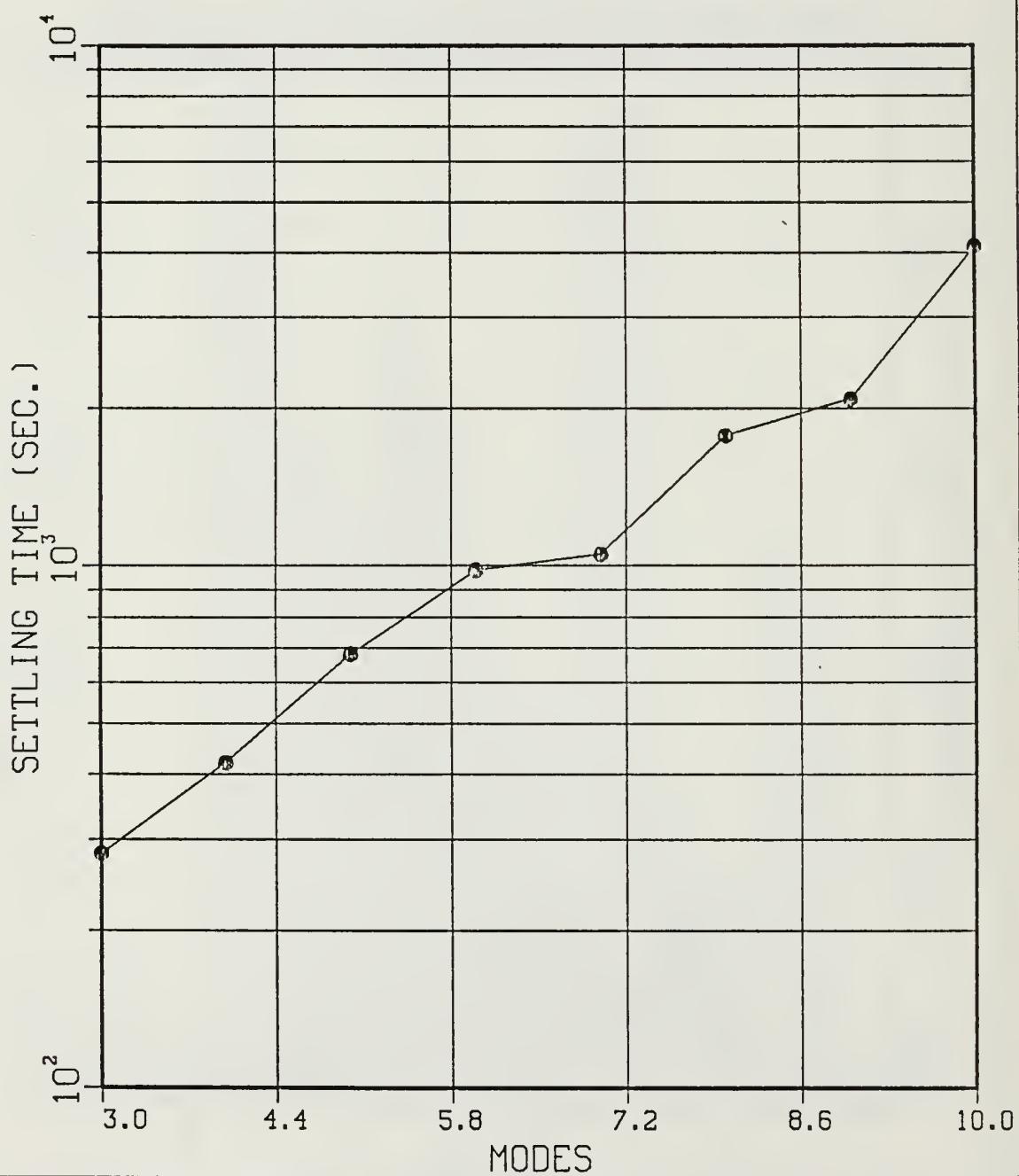


Figure 16. Settling Time versus number of Modes Observed

V. CONCLUSIONS AND RECOMMENDATIONS

A. CONCLUSIONS

Simulations runs showed that a matched plant/observer can work if the following criterions are meet:

- The ratio of plant noise to measurement noise is sufficiently high to produce a usable settling time.
- Sufficient computational power is available to run the matched observer. The amount of memory and number of computation goes up as the number of modes observed increases.

Utilizing a reduce order observer for an arbitrarily selected set of modes is not feasible. The non-observed modes add so much noise to the system that settling times and observer performance are so poor as to render the observer useless for obtaining state values for plant control.

B. RECOMENDATIONS

The work on the Kalman Observer for Large Space Structures lead to the following recommendations for further research:

- Identify those modes that contribute the largest noise to the Kalman observer and set the observer to estimate these states in addition to those required for plant control. A possible method for identifying the modes that contribute the largest noise to the observer would be the Karhunen-Loeve expansion.
- Modifying the plant/observer to model the use of sensors at additional positions to see if the increase in the data rate will help decrease settling time.
- Modify the model to incorporate noise injection into more than one location. The current model has noise injected at only one position, a useful simplification for initial analysis but not realistic.

APPENDIX A. KALMAN GAIN MATRIX GENERATION PROGRAM

```

C ***** GAIN ***** GMA00010
C ***** ***** GMA00020
C ***** GAIN ***** GMA00030
C ***** ADAPTED TO RUN KALMAN FILTER AND COMPUTE THE GMA00040
C ***** G MATRIX BY ITERATION STOPPING WHEN THE GMA00050
C ***** THE MATRIX GOES TO STEADY STATE GMA00060
C ***** ***** ***** ***** ***** ***** ***** ***** ***** GMA00070
C ***** ***** ***** ***** ***** ***** ***** ***** ***** GMA00080
C ***** ***** ***** ***** ***** ***** ***** ***** ***** GMA00090
C ***** ***** ***** ***** ***** ***** ***** ***** ***** GMA00100
C ***** VARIABLE DECLARATIONS ***** ***** ***** ***** GMA00110
C ***** ***** ***** ***** ***** ***** ***** ***** ***** GMA00120
C ***** ***** ***** ***** ***** ***** ***** ***** ***** GMA00130
C
C EXTERNAL STMTRX,DLINRG,EXCMS, DEVCRG GMA00140
C CHARACTER*6 NAM GMA00150
C CHARACTER*1 AGAIN, CORECT, RAGAIN GMA00160
C INTEGER ROWN1, ROWN2, ROWN3, COUNT, NODE, MODE, KQ, EMODE, SMODE, R2M, C2M GMA00170
C INTEGER CT, CF, KADJ, CFADJ, LOOP, PRNT, JJ, JK, N1, JR, KR, MR, ISEED, M2 GMA00180
C INTEGER JL, J1, JM GMA00190
C REAL LAMA(100), UGVEX(684, 100), RNODE, RMODE, MIN GMA00200
C REAL*8 PHI(2, 2, 100), GAMMA(2, 100), EGT, GMA, WN, W1, X1T, X2T, TIME GMA00210
C REAL*8 A(200, 200), B(200, 3), F(3, 50), IMPLSE, ENERGY GMA00220
C REAL*8 COSW1T, SINW1T, X1(100), X2(100), COST(100) GMA00230
C REAL*8 DAMP, SAMPT, PI, SAMPTM, SUM1, SUM2, SUM3, SUMC GMA00240
C REAL*8 C(6, 200), IDENT(50, 50), RMN(6, 6), QPN(3, 3) GMA00250
C REAL*8 PK(50, 50), Y(6), BN(200, 3) GMA00260
C REAL*8 PNVARX, PNVARY, PNVARZ GMA00270
C REAL*8 MNVX1, MNVY1, MNVZ1, SUM, BQBT(50, 50) GMA00280
C REAL*8 TMP1(50, 3), TMP2(3, 3), TMP3(50, 50) GMA00290
C REAL*8 PK1(50, 50), G(50, 3) GMA00300
C REAL*8 DY(3), ES, ED, ESUM, CGN, PRT GMA00310
C REAL*8 SF, N9, TCHK, ACHK, H1, H2, H3, H4, H5, H6 GMA00320
C REAL*8 AGC(100, 100) GMA00330
C
C COMPLEX*8 EVAL(100), EVEC (100, 100) GMA00340
C
C ***** ***** ***** ***** ***** ***** ***** ***** ***** GMA00350
C ***** ***** ***** ***** ***** ***** ***** ***** ***** GMA00360
C ***** ***** ***** ***** ***** ***** ***** ***** ***** GMA00370
C ***** ***** ***** ***** ***** ***** ***** ***** ***** GMA00380
C ***** ***** ***** ***** ***** ***** ***** ***** ***** GMA00390
C ***** VARIABLE DEFINITIONS ***** ***** ***** ***** GMA00400
C ***** ***** ***** ***** ***** ***** ***** ***** ***** GMA00410
C ***** ***** ***** ***** ***** ***** ***** ***** ***** GMA00420
C STMTRX = SUBROUTINE ESTABLISHES STATE TRANSITION MATRICES GMA00430
C LAMA = VECTOR OF THE SQUARE OF THE NATURAL FREQUENCIES GMA00440
C UGVEX = MODE POSITIONS AND SLOPES OF THE NODAL POINTS GMA00450
C PHI = STATE TRANSITION MATRICES FOR EACH MODE GMA00460
C GAMMA = INPUT TRANSITION MATRIX GMA00470
C A = DIAGONAL MATRIX CONSISTING OF PHI GMA00480
C B = INPUT MATRIX OF GAMMA AND CONTROL SLOPES GMA00490
C DAMP = DAMPING FACTOR GMA00500
C SAMPT = SAMPLING TIME GMA00510

```

```

C TCX, TCY, TCZ = CONTROL TORQUE VALUES GMA00520
C ENERGY = TOTAL SYSTEM ENERGY GMA00530
C IMPLSE = IMPULSE INPUT FUNCTION GMA00540
C MIN = NUMBER OF MINUTES SYSTEM WILL BE OBSERVED GMA00550
C SMODE = NUMBER OF STARTING MODE (INT) GMA00560
C MODE = NUMBER OF MODES (INT) GMA00570
C EMODE = NUMBER OF THE LAST MODE (INT) GMA00580
C NODE = NUMBER OF THE NOISE INPUT MODE (INT) GMA00590
C *** NOISE SLOPE LOCATIONS IN DATA MATRIX ***
C ROWN1 = X-SLOPE LOCATION GMA00600
C ROWN2 = Y-SLOPE LOCATION GMA00610
C ROWN3 = Z-SLOPE LOCATION GMA00620
C C = OUTPUT MATRIX FOR Y GMA00630
C IDENT = IDENTITY MATRIX GMA00640
C RMN = MEASUREMENT NOISE COVARIANCE MATRIX GMA00650
C QPN = PLANT NOISE COVARIANCE MATRIX GMA00660
C PNVARX = PLANT NOISE X-SLOPE VARIANCE GMA00670
C PNVARY = PLANT NOISE Y-SLOPE VARIANCE GMA00680
C PNVARZ = PLANT NOISE Z-SLOPE VARIANCE GMA00690
C MNVARX = MEASUREMENT NOISE X-SLOPE VARIANCE GMA00700
C MNVARY = MEASUREMENT NOISE Y-SLOPE VARIANCE GMA00710
C MNVARZ = MEASUREMENT NOISE Z-SLOPE VARIANCE GMA00720
C ISEED = INITIALIZATION FOR RANDOM NUMBER GENERATOR GMA00730
C XKAL = X MATRIX GMA00740
C Y = OUTPUT MATRIX GMA00750
C RNDM = RANDOM NUMBERS USED FOR WHITE NOISE IN MEASUREMENTS AND GMA00760
C IN PLANT FORCES GMA00770
C BN = B MATRIX TO MULTIPLY NOISE DISTURBANCES GMA00780
C TNX,TNY,TNZ= NOISE TORQUES X,Y,Z SLOPES GMA00790
C M2=2*MODE GMA00800
C
C
C *****
C
C ***** SAMPLE OF SPACE EXEC FILE *****
C
C THIS FILE MUST BEGIN IN COLUMN 1 AND RUN WITH THE FOLLOWING GMA00850
C SEQUENCE FOR THE INITIAL RUN OF THE PROGRAM: GMA00860
C
C FORTVS SPACE (COMPILES PROGRAM) GMA00870
C SPACE (EXECUTES EXEC FILE) GMA00880
C LOAD SPACE (START (LOADS AND EXECUTES PROGRAM) GMA00890
C
C SUBSEQUENT PROGRAM RUNS CAN ELIMINATE "FORTVS SPACE" IF NO GMA00940
C CHANGES HAVE BEEN MADE TO THE PROGRAM, AND CAN ELIMINATE GMA00950
C RUNNING THE EXEC FILE. GMA00960
C
C FI 4 DISK THESIS INPUT B (PERM GMA00970
C FI 8 DISK UTILITY DATA (RECFM VS BLOCK 133 PERM GMA00980
C FI 11 DISK CNTRL OUTPUT (RECFM F BLOCK 80 LRECL 80 PERM GMA00990
C FI 13 DISK GAMMA OUTPUT (RECFM VS BLOCK 133 PERM GMA01000
C FI 14 DISK MODE OUTPUT (RECFM F BLOCK 80 LRECL 80 PERM GMA01010
C FI 16 DISK COST OUTPUT (RECFM F BLOCK 80 LRECL 80 PERM GMA01020
C FI 17 DISK PRT OUTPUT (RECFM F BLOCK 80 LRECL 80 PERM GMA01030
C FI 18 DISK ERROR DATA (RECFM F BLOCK 80 LRECL 80 PERM GMA01040
C FI 19 DISK END FILE (RECFM F BLOCK 80 LRECL 80 PERM GMA01050
C FI 20 DISK GMAT FILE (RECFM F BLOCK 80 LRECL 80 PERM GMA01060
C

```

```

C GMA01080
C GMA01090
C GMA01100
C GMA01110
C GMA01120
C GMA01130
C GMA01140
C GMA01150
C GMA01160
C GMA01170
C GMA01180
C GMA01190
C GMA01200
C GMA01210
C GMA01220
C GMA01230
C GMA01240
C GMA01250
C GMA01260
C GMA01270
C GMA01280
C GMA01290
C GMA01300
C GMA01310
C GMA01320
C GMA01330
C GMA01340
C GMA01350
C GMA01360
C GMA01370
C GMA01380
C GMA01390
C GMA01400
C GMA01410
C GMA01420
C GMA01430
C GMA01440
C GMA01450
C GMA01460
C GMA01470
C GMA01480
C GMA01490
C GMA01500
C GMA01510
C GMA01520
C GMA01530
C GMA01540
C GMA01550
C GMA01560
C GMA01570
C GMA01580
C GMA01590
C GMA01600
C GMA01610
C GMA01620

***** PARAMETER ( JR=5243, KR=5397, MR=262139) *****
C
C MIN =1200.0
C WT=1.0D00
C PI = 4.0D0 * ATAN(1.0D0)
C
C ***** READ LAMA AND UGVEX MATRICIES *****
C
C CALL EXCMS ('CLRSCRN')
C WRITE(6,1008)
C WRITE(6,*) ' '
C READ(4,1001) NAM
C READ(4,1002)(LAMA(I),I=1,100)
C READ(4,1001) NAM
C DO 5 J = 1,100
C     READ(4,1002)(UGVEX(I,J),I=1,684)
C 5 CONTINUE
C
1001 FORMAT(1X,A6)
1002 FORMAT(1X,8E15.8)
1008 FORMAT(1X,///)
C
500 CALL EXCMS ('CLRSCRN')
C
C ***** STARTING MODE NUMBER *****
C ** SMODE 7 TO 100 (INTEGER) *****
C SMODE=10
C
700 WRITE (16,700) SMODE
C FORMAT (' ','STARTING MODE NUMBER: ',I2)
C
C ***** NUMBER OF MODES TO SCAN *****
C ** MODE 1 TO 93 (INTEGER) *****
C
C MODE= 3
C
C EMODE = SMODE + MODE - 1
C
C WRITE (16,701) MODE
701 FORMAT (' ','NUMBER OF MODES SCANNED: ',I2)
C
C ***** NOISE INPUT POSITION *****
C ** NODE 1 TO 114 (INTEGER) (IF 0 THEN NO NOISE INPUT) *****
C NODE= 8
C

```

```

      WRITE (16,702) NODE          GMA01630
702  FORMAT (' ','NOISE NODE LOCATION: ',I5)  GMA01640
C
C
C      *****          SAMPLING TIME          *****
C      ** SAMPT MUST BE LESS THAN OR EQUAL TO SAMPTM **
C      SAMPT = .05
      SAMPTM = ((2.0D0*PI)/SQRT(LAMA(EMODE)))/2.0D0
      IF (SAMPT.GE.SAMPTM) THEN
          SAMPT=SAMPTM
      ENDIF
C
      WRITE (16,900) MIN          GMA01650
900  FORMAT (' ',2X,'MIN: ',F8.3)  GMA01660
C
      WRITE (16,703) SAMPT         GMA01670
703  FORMAT (' ','SAMPLING TIME: ',D12.4)  GMA01680
C
C      *****          DAMPING FACTOR          *****
C      ** DAMP 0.0 TO 1.0 (REAL*8)
C      DAMP=.01
C
      WRITE (16,704) DAMP         GMA01700
704  FORMAT (' ','DAMPING FACTOR: ',D12.4)  GMA01710
C
C
C      *** PLANT NOISE VARIANCE ***
C      ** PNVARX, PNVARY, PNVARZ GT 0.0
C
      SF1=2.5D06
C
      PNVARX=1.0D00*SF1
      PNVARY=1.0D00*SF1
      PNVARZ=1.0D00*SF1
C
C
C      *** MEASUREMENT NOISE VARIANCE ***
C      ** MNVARX, MNVARY, MNVARZ GT 0.0
      SF=1.0
      MNVX1=5.5D-03*SF
      MNVY1=5.5D-03*SF
      MNVZ1=5.5D-03*SF
C
510  CALL EXCMS ('CLRSCRN')
      WRITE (6,1008)
      WRITE (6,*) '                                PROGRAM RUNNING'
C
C      *****          NOISE INPUT LOCATION          *****
C
      ROWN3 = NODE*6
      ROWN2 = (NODE*6) - 1
      ROWN1 = (NODE*6) - 2
      COUNT = 0

```

```

C *****          INITIALIZE MATRICIES *****          GMA02180
C
C      DO 40 I = 1,3
C          DO 45 J = 1,3
C              RMN(I,J)=0.0
C          CONTINUE
45      CONTINUE
40
C      DO 47 I=1,50
C          DO 46 J=1,50
C              IDENT(I,J)=0.0
C              PK(I,J)=0.0
C          CONTINUE
46      CONTINUE
47      CONTINUE
C
C      DO 48 K=1,50
C          IDENT(K,K)=1.0
48      CONTINUE
C
C      *** INITIALIZE RMN AND QPN MATRICES ***
C
C      DO 60 I=1,3
C          DO 58 J=1,3
C              QPN(I,J)=0.0
C          CONTINUE
58      CONTINUE
60      CONTINUE
C
C      RMN(1,1)=MNVX1***2
C      RMN(2,2)=MNVY1***2
C      RMN(3,3)=MNVZ1***2
C      QPN(1,1)=PNVARX***2.0
C      QPN(2,2)=PNVARY***2.0
C      QPN(3,3)=PNVARZ***2.0
C
C 9999  FORMAT (' ',',',',')
C *****          BEGIN MAIN PROGRAM *****          GMA02530
C
C      *** PRE-LOOP PORTION OF KALMAN FILTER
C      JK=SMODE*2-2
C      M2=2*MODE
C      DO 94 I=1,3
C          DO 92 J=1,M2
C              JL=JK+J
C              SUM=0.0
C                  DO 90 K=1,3
C                      SUM=SUM+QPN(I,K)*BN(JL,K)
C                  CONTINUE
90          TMP1(J,I)=SUM
C          CONTINUE
92          CONTINUE
94          CONTINUE

```

```

C                                         GMA02740
C                                         GMA02750
DO 98 I=1,M2                                         GMA02760
JL=JK+I                                         GMA02770
DO 97 J=1,M2                                         GMA02780
SUM=0.0                                         GMA02790
DO 96 K=1,3                                         GMA02800
SUM=SUM+BN(JL,K)*TMP1(J,K)                         GMA02810
96      CONTINUE                                     GMA02820
BQBT(I,J)=SUM                                     GMA02830
97      CONTINUE                                     GMA02840
98      CONTINUE                                     GMA02850
C                                         GMA02860
M2=2*MODE                                         GMA02870
DO 100 I=1,M2                                         GMA02880
DO 99 J=1,M2                                         GMA02890
TMP3(I,J)=0.0                                     GMA02900
99      CONTINUE                                     GMA02910
100     CONTINUE                                     GMA02920
JL=JK+M2                                         GMA02930
DO 9375 I=1,3                                         GMA02940
DO 9374 J=1,JL                                         GMA02950
C(I,J)=C(I,J)*SF                                     GMA02960
9374     CONTINUE                                     GMA02970
9375     CONTINUE                                     GMA02980
C                                         GMA02990
C ***** THIS SECTION COMPUTES THE STATE UPDATE ***** GMA03000
C ***** THIS SECTION COMPUTES THE STATE UPDATE ***** GMA03010
C ***** THIS SECTION COMPUTES THE STATE UPDATE ***** GMA03020
C                                         GMA03030
ESUM=0.0                                         GMA03040
COUNT = 0                                         GMA03050
ENERGY = 0.0DO                                     GMA03060
TIME = 0.0                                         GMA03070
CGN=0.0                                         GMA03080
C ***** SETS LOOP FOR THE ITERATIONS NECESSARY TO OBSERVE ***** GMA03090
C ***** THE SYSTEM FOR THE NUMBER OF MINUTES SPECIFIED ***** GMA03100
C                                         GMA03110
LOOP = INT((MIN*60.0)/SAMPT)                         GMA03120
PRT=(DBLE(LOOP))/1200.0                               GMA03130
PTA=(DBLE(LOOP))/2400.0                               GMA03140
CNTA=0.0                                         GMA03150
ACHK=1.0D-10                                         GMA03160
H1=0.0                                         GMA03170
H2=0.0                                         GMA03180
H3=0.0                                         GMA03190
H4=0.0                                         GMA03200
H5=0.0                                         GMA03210
H6=0.0                                         GMA03220
TCHK=MIN*60.0                                         GMA03230
9991    CONTINUE                                     GMA03240
C                                         GMA03250
TIME = TIME+ SAMPT                                 GMA03260
C                                         GMA03270
CGN=CGN+1.0                                         GMA03280
C                                         GMA03290
CNTA=CNTA+1.0

```

```

C      *** START OF KALMAN FILTER ***
C
C      M2=2*MODE
C
C      *** COMPUTATION OF PK*AT ***
C
C      JK=2*SMODE-2
C      DO 175 I=1,M2
C          DO 170 J=1,M2
C              JL=JK+J
C              SUM=0.0
C                  DO 165 K=1,M2
C                      JM=JK+K
C                      SUM =SUM+PK(I,K)*A(JL,JM)
C
165      CONTINUE
C          TMP3(I,J)=SUM
C
170      CONTINUE
C
175      CONTINUE
C
C      *** COMPUTATION OF A*(PK*AT)+ BQBT = PK1 ***
C
C      DO 190 I=1,M2
C          JL=JK+I
C              DO 185 J=1,M2
C                  SUM=0.0
C                      DO 180 K=1,M2
C                          JM=JK+K
C                          SUM=SUM+A(JL,JM)*TMP3(K,J)
C
180      CONTINUE
C          PK1(I,J)=SUM+BQBT(I,J)
C
185      CONTINUE
C
190      CONTINUE
C
C      *****
C
C      *** COMPUTE PK1*CT *****
C
C      DO 205 I=1,M2
C          DO 200 J=1,3
C              SUM=0.0
C                  DO 195 K=1,M2
C                      JM=JK+K
C                      SUM=SUM+PK1(I,K)*C(J,JM)
C
195      CONTINUE
C          TMP1(I,J)=SUM
C
200      CONTINUE
C
205      CONTINUE
C
C      *****
C
C      *** COMPUTE C*(PK1*CT)+RMN ***
C
C      DO 220 I=1,3
C          DO 215 J=1,3
C              SUM=0.0
C                  DO 210 K=1,M2
C                      JM=JK+K

```

```

      SUM=SUM+C(I,JM)*TMP1(K,J)          GMA03840
210    CONTINUE
      TMP2(I,J)=SUM+RMN(I,J)          GMA03850
215    CONTINUE
220    CONTINUE
C
C      *** COMPUTATION OF THE INVERSE OF C*PK1*CT + R
C
C      CALL DLINRG ( 3,TMP2,3,TMP2,3)          GMA03860
C
C      *** COMPUTE CT*INV(C*PK1*CT+R)          GMA03870
C
      DO 245 I=1,M2          GMA03880
      JL=JK+I          GMA03890
         DO 240 J=1,3          GMA03900
         SUM=0.0          GMA03910
            DO 235 K=1,3          GMA03920
            SUM=SUM+C(K,JL)*TMP2(K,J)          GMA03930
235      CONTINUE          GMA03940
      TMP1(I,J)=SUM          GMA03950
240      CONTINUE          GMA03960
245      CONTINUE          GMA03970
C      ****
C
C      *** COMPUTE PK1*C*INV(C*PK1*CT+R) = G *****
C
      DO 260 I=1,M2          GMA04000
         DO 255 J=1,3          GMA04010
         SUM=0.0          GMA04020
            DO 250 K=1,M2          GMA04030
            SUM=SUM+PK1(I,K)*TMP1(K,J)          GMA04040
250      CONTINUE          GMA04050
      G(I,J)=SUM          GMA04060
255      CONTINUE          GMA04070
260      CONTINUE          GMA04080
C
      N9=DABS((G(1,1)-H1)/G(1,1))          GMA04090
      IF (N9.GT.ACHK)      THEN          GMA04100
      GO TO 7377          GMA04110
      END IF          GMA04120
      N9=DABS((G(1,3)-H2)/G(1,3))          GMA04130
      IF (N9.GT.ACHK)THEN          GMA04140
      GO TO 7377          GMA04150
      END IF          GMA04160
      N9=DABS((G(2,1)-H3)/G(2,1))          GMA04170
      IF (N9.GT.ACHK)      THEN          GMA04180
      GO TO 7377          GMA04190
      END IF          GMA04200
      N9=DABS((G(2,3)-H4)/G(2,3))          GMA04210
      IF (N9.GT.ACHK)      THEN          GMA04220
      GO TO 7377          GMA04230
      END IF          GMA04240
      N9=DABS((G(3,3)-H5)/G(3,3))          GMA04250
      IF (N9.GT.ACHK)      THEN          GMA04260
      GO TO 7377          GMA04270
      END IF          GMA04280
      N9=DABS((G(1,1)-H1)/G(1,1))          GMA04290
      IF (N9.GT.ACHK)      THEN          GMA04300
      GO TO 7377          GMA04310
      END IF          GMA04320
      N9=DABS((G(1,3)-H2)/G(1,3))          GMA04330
      IF (N9.GT.ACHK)      THEN          GMA04340
      GO TO 7377          GMA04350
      END IF          GMA04360
      N9=DABS((G(2,1)-H3)/G(2,1))          GMA04370
      IF (N9.GT.ACHK)      THEN          GMA04380
      GO TO 7377          GMA04390

```

```

END IF GMA04400
N9=DABS((G(M2,3)-H6)/G(M2,3)) GMA04410
IF (N9.GT. ACHK) THEN GMA04420
GO TO 7377 GMA04430
END IF GMA04440
GO TO 400 GMA04450
C GMA04460
C GMA04470
7377 CONTINUE GMA04480
H1=G(1,1) GMA04490
H2=G(1,3) GMA04500
H3=G(2,1) GMA04510
H4=G(2,3) GMA04520
H5=G(3,3) GMA04530
H6=G(M2,3) GMA04540
GMA04550
GMA04560
GMA04570
IF (TCHK.LE. TIME) THEN GMA04580
GO TO 400 GMA04590
END IF GMA04600
IF (CGN.GE. PRT) THEN GMA04610
C GMA04620
WRITE (6,*) 'TIME= ', TIME, ' SEC.' GMA04630
C GMA04640
WRITE (6,*) 'N9= ', N9 GMA04650
CGN=0.0 GMA04660
END IF GMA04670
C GMA04680
C **** COMPUTE IDENT - G*C GMA04690
C GMA04700
DO 275 I=1,M2 GMA04710
  DO 270 J=1,M2 GMA04720
    JL=JK+J GMA04730
    SUM=0.0 GMA04740
      DO 265 K=1,3 GMA04750
        SUM=SUM+G(I,K)*C(K,JL) GMA04760
    CONTINUE GMA04770
265  TMP3(I,J)= IDENT(I,J)-SUM GMA04780
270  CONTINUE GMA04790
275  CONTINUE GMA04800
C **** GMA04810
C GMA04820
C **** COMPUTE PK= (IDENT - G*C)*PK1 GMA04830
C GMA04840
DO 290 I=1,M2 GMA04850
  DO 285 J=1,M2 GMA04860
    SUM=0.0 GMA04870
      DO 280 K=1,M2 GMA04880
        SUM=SUM+TMP3(I,K)*PK1(K,J) GMA04890
    CONTINUE GMA04900
280  PK(I,J)=SUM GMA04910
285  CONTINUE GMA04920
290  CONTINUE GMA04930
C **** GMA04940
C GMA04950

```

```

C      END OF KALMAN FILTER PART 1 - START OF PART 2 *****
C
C
C      GO TO 9991
C
400    CONTINUE
C
      WRITE (20,1008)
      WRITE (20,*) 'TIME= ',TIME
      DO 384 I=1,M2
      WRITE (20,5350)    G(I,1),G(I,2),G(I,3)
384    CONTINUE
5350    FORMAT (' ',5X,D15.8 ,5X,D15.8 ,5X,D15.8 )
      WRITE (20,*) 'N9= ',N9
C
C      **** COMPUTE AGC = A - G*C
C
      M2=2*MODE
      JK=2*SMODE-2
C
      DO 7155 I=1,M2
      JL=JK+I
      DO 7154 J=1,M2
      JM=JK+J
      SUM=0.0
      DO 7153 K=1,3
      SUM=SUM+G(I,K)*C(K,JM)
7153    CONTINUE
      AGC(I,J)=A(JL,JM)-SUM
7154    CONTINUE
7155    CONTINUE
C
C
C      *** COMPUTE THE EIGENVALUES OF AGC
C
      CALL DEVCRG (M2, AGC, 100, EVAL, EVEC, 100)
C
C      **** PRINT EVAL (EIGENVALUE) MATRIX
C
      DO 7157 I=1,M2
      WRITE (20,*) 'I= ', I, 'EIG= ', EVAL(I)
7157    CONTINUE
C
C
C
599    STOP
      END
C
C
C
C
C      **** THIS SUBROUTINE COMPUTES THE STATE TRANSITION MATRIX FOR EACH
C
      **** GMA04960
      **** GMA04970
      **** GMA04980
      **** GMA04990
      **** GMA05000
      **** GMA05010
      **** GMA05020
      **** GMA05030
      **** GMA05040
      **** GMA05050
      **** GMA05060
      **** GMA05070
      **** GMA05080
      **** GMA05090
      **** GMA05100
      **** GMA05110
      **** GMA05120
      **** GMA05130
      **** GMA05140
      **** GMA05150
      **** GMA05160
      **** GMA05170
      **** GMA05180
      **** GMA05190
      **** GMA05200
      **** GMA05210
      **** GMA05220
      **** GMA05230
      **** GMA05240
      **** GMA05250
      **** GMA05260
      **** GMA05270
      **** GMA05280
      **** GMA05290
      **** GMA05300
      **** GMA05310
      **** GMA05320
      **** GMA05330
      **** GMA05340
      **** GMA05350
      **** GMA05360
      **** GMA05370
      **** GMA05380
      **** GMA05390
      **** GMA05400
      **** GMA05410
      **** GMA05420
      **** GMA05430
      **** GMA05440
      **** GMA05450
      **** GMA05460
      **** GMA05470
      **** GMA05480
      **** GMA05490
      **** GMA05500
      **** GMA05510

```

```

C OF THE 100 MODES
C ****
C
C SUBROUTINE STMTRX(EMODE,SMODE,T,D,PHI,GAMMA,A,B,LAMA,UGVEX,C,
+      ROWN1,ROWN2,ROWN3,BN)
C
C REAL*8 WN,GMA,PHI(2,2,100),GAMMA(2,100),EGT,T,COSW1T,SINW1T
C REAL*8 W1,D,A(200,200),B(200,3),C(6,200),BN(200,3)
C REAL LAMA(100),UGVEX(684,100)
C INTEGER SMODE,R,EMODE,JJ,KK,ROWN1,ROWN2,ROWN3
C
C DO 600 I = 1      ,100
C     WN = DBLE(SQRT(LAMA(I)))
C     GMA = D*WN/2.0
C     EGT = DEXP(-GMA*T)
C     W1 = DSQRT((WN**2)-(GMA**2))
C     COSW1T = DCOS(W1*T)
C     SINW1T = DSIN(W1*T)
C
C IF(WN.EQ.0)THEN
C     PHI(1,1,I) = EGT*COSW1T
C     PHI(1,2,I) = T
C     PHI(2,1,I) = 0
C     PHI(2,2,I) = EGT*COSW1T
C
C     GAMMA(1,I) = 0
C     GAMMA(2,I) = 0
C ELSE
C
C     PHI(1,1,I) = EGT*(COSW1T + (GMA*(W1**(-1)))*SINW1T)
C     PHI(1,2,I) = (W1**(-1))*EGT*SINW1T
C     PHI(2,1,I) = -(WN**2)*(W1**(-1))*EGT*SINW1T
C     PHI(2,2,I) = EGT*(COSW1T - (GMA*(W1**(-1)))*SINW1T)
C
C GAMMA(1,I)=(WN**(-2))*(1.0-EGT*COSW1T-EGT*(GMA/W1)*SINW1T)
C
C ****

```



```

BN(R+1,3)=GAMMA(2,K)*DBLE(UGVEX(ROWN3,K)) GMA06630
C GMA06640
C GMA06650
C GMA06660
C GMA06670
C GMA06680
C GMA06690
C GMA06700
R = R+2 GMA06710
610 CONTINUE GMA06720
C GMA06730
C GMA06740
C GMA06750
C GMA06760
C GMA06770
C GMA06780
C **** C MATRIX PRODUCTION **** GMA06790
C GMA06800
C GMA06810
C GMA06820
JJ=-1 GMA06830
DO 640 I=1,100 GMA06840
JJ=JJ+1 GMA06850
KK=I+JJ GMA06860
C GMA06870
C GMA06880
C C(1,KK) = DBLE(UGVEX(418,I)) GMA06890
C(2,KK) = DBLE(UGVEX(419,I)) GMA06900
C(3,KK) = DBLE(UGVEX(420,I)) GMA06910
C GMA06920
C GMA06930
C GMA06940
KK=KK+1 GMA06950
C GMA06960
C C(1,KK)=0.0 GMA06970
C(2,KK)=0.0 GMA06980
C(3,KK)=0.0 GMA06990
C GMA07000
640 CONTINUE GMA07010
C GMA07020
C GMA07030
C GMA07040
RETURN GMA07050
END GMA07060

```

APPENDIX B. KALMAN OBSERVER AND PLANT SIMULATION

```

C ***** SIMRUN ***** SIM00010
C ***** ADAPTED TO READ KALMAN FILETER G MATRICE ***** SIM00020
C ***** THEN RUN ALL N MODES OF THE PLANT WHILE ***** SIM00030
C ***** USING A KALMAN FILTER TO OBSERVE M ***** SIM00040
C ***** NUMBER OF STATES ***** SIM00050
C ***** ***** ***** ***** ***** ***** ***** ***** SIM00060
C ***** ***** ***** ***** ***** ***** ***** ***** SIM00070
C ***** ***** ***** ***** ***** ***** ***** ***** SIM00080
C ***** ***** ***** ***** ***** ***** ***** ***** SIM00090
C ***** ***** ***** ***** ***** ***** ***** ***** SIM00100
C ***** ***** ***** ***** ***** ***** ***** ***** SIM00110
C ***** VARIABLE DECLARATIONS ***** SIM00120
C ***** ***** ***** ***** ***** ***** ***** ***** SIM00130
C ***** ***** ***** ***** ***** ***** ***** ***** SIM00140
C EXTERNAL STMTRX,EXCMS
CHARACTER*6 NAM
CHARACTER*1 AGAIN, CORECT, RAGAIN
INTEGER ROWN1, ROWN2, ROWN3, COUNT, NODE, MODE, KQ, EMODE, SMODE, R2M, C2M
INTEGER CT, CF, KADJ, CFADJ, LOOP, PRNT, JJ, JK, N1, JR, KR, MR, ISEED, M2
INTEGER ITYPE(200), IPVT(100), NS, NF, SN, FN
INTEGER JL, J1, JM, JP, JQ, KA, KB, KC, KD, KE, KF, KG
C
C
C
REAL LAMA(100), UGVEX(684,100), RNODE, RMODE, MIN
REAL*8 PHI(2,2,100), GAMMA(2,100), EGT, GMA, WN, W1, X1T, X2T, TIME
REAL*8 A(200,200), B(200,3), F(3, 50), IMPLSE, ENERGY
REAL*8 COSW1T, SINW1T, X(200)
REAL*8 TCX, TCY, TCZ, DAMP, SAMPT, PI, SAMPTM, SUM1, SUM2, SUM3, SUMC
REAL*8 C(3,200), IDENT( 50, 50), RMN(3,3), QPN(3,3)
REAL*8 Y(3) , BN(200,3)
REAL*8 PNVARX, PNVARY, PNVARZ
REAL*8 MNVARX, MNVARY, MNVARZ
REAL*8 SUM, RNDM(6), RND1, RND2
REAL*8 XH( 50) , BQBT( 50, 50)
REAL*8 SF1
REAL*8 TMP1( 50,3), TMP2(3,3), TMP3( 50, 50)
REAL*8 G( 50,3)
REAL*8 XH1( 50) , DY(3) , ES, ED, ESUM, CGN, PRT
REAL*8 WT , WD(3), BNWD(200)
REAL*8 AX(200) , V(3), SF , TO, CTT, ESS
REAL*8 CTG, XDEL, E2(100), XDEL1, ERS, PRT1, E3(100), XS(100)
C ***** ***** ***** ***** ***** ***** ***** ***** SIM00250
C ***** ***** ***** ***** ***** ***** ***** ***** SIM00260
C ***** ***** ***** ***** ***** ***** ***** ***** SIM00270
C ***** ***** ***** ***** ***** ***** ***** ***** SIM00280
C ***** ***** ***** ***** ***** ***** ***** ***** SIM00290
C ***** ***** ***** ***** ***** ***** ***** ***** SIM00300
C ***** ***** ***** ***** ***** ***** ***** ***** SIM00310
C ***** ***** ***** ***** ***** ***** ***** ***** SIM00320
C ***** ***** ***** ***** ***** ***** ***** ***** SIM00330
C ***** ***** ***** ***** ***** ***** ***** ***** SIM00340
C ***** ***** ***** ***** ***** ***** ***** ***** SIM00350
C ***** ***** ***** ***** ***** ***** ***** ***** SIM00360
C ***** ***** ***** ***** ***** ***** ***** ***** SIM00370
C ***** ***** ***** ***** ***** ***** ***** ***** SIM00380
C ***** ***** ***** ***** ***** ***** ***** ***** SIM00390
C ***** ***** ***** ***** ***** ***** ***** ***** SIM00400
C ***** ***** ***** ***** ***** ***** ***** ***** SIM00410
C ***** ***** ***** ***** ***** ***** ***** ***** SIM00420
C ***** ***** ***** ***** ***** ***** ***** ***** SIM00430
C ***** ***** ***** ***** ***** ***** ***** ***** SIM00440
C ***** VARIABLE DEFINITIONS ***** SIM00450
C ***** ***** ***** ***** ***** ***** ***** ***** SIM00460
C ***** ***** ***** ***** ***** ***** ***** ***** SIM00470
C STMTRX = SUBROUTINE ESTABLISHES STATE TRANSITION MATRICES SIM00480
C LAMA = VECTOR OF THE SQUARE OF THE NATURAL FREQUENCIES SIM00490
C UGVEX = MODE POSITIONS AND SLOPES OF THE NODAL POINTS SIM00500
C PHI = STATE TRANSITION MATRICES FOR EACH MODE SIM00510

```

```

C GAMMA = INPUT TRANSITION MATRIX SIM00520
C A = DIAGONAL MATRIX CONSISTING OF PHI SIM00530
C B = INPUT MATRIX OF GAMMA AND CONTROL SLOPES SIM00540
C DAMP = DAMPING FACTOR SIM00550
C SAMPT = SAMPLING TIME SIM00560
C TCX, TCY, TCZ = CONTROL TORQUE VALUES SIM00570
C ENERGY = TOTAL SYSTEM ENERGY SIM00580
C IMPLSE = IMPULSE INPUT FUNCTION SIM00590
C MIN = NUMBER OF MINUTES SYSTEM WILL BE OBSERVED SIM00600
C SMODE = NUMBER OF STARTING MODE (INT) SIM00610
C MODE = NUMBER OF MODES (INT) SIM00620
C EMODE = NUMBER OF THE LAST MODE (INT) SIM00630
C NODE = NUMBER OF THE NOISE INPUT MODE (INT) SIM00640
C *** NOISE SLOPE LOCATIONS IN DATA MATRIX ***
C ROWN1 = X-SLOPE LOCATION SIM00660
C ROWN2 = Y-SLOPE LOCATION SIM00670
C ROWN3 = Z-SLOPE LOCATION SIM00680
C C = OUTPUT MATRIX FOR Y SIM00690
C IDENT = IDENTITY MATRIX SIM00700
C RMN = MEASUREMENT NOISE COVARIANCE MATRIX SIM00710
C QPN = PLANT NOISE COVARIANCE MATRIX SIM00720
C PNVARX = PLANT NOISE X-SLOPE VARIANCE SIM00730
C PNVARY = PLANT NOISE Y-SLOPE VARIANCE SIM00740
C PNVARZ = PLANT NOISE Z-SLOPE VARIANCE SIM00750
C MNVARX = MEASUREMENT NOISE X-SLOPE VARIANCE SIM00760
C MNVARY = MEASUREMENT NOISE Y-SLOPE VARIANCE SIM00770
C MNVARZ = MEASUREMENT NOISE Z-SLOPE VARIANCE SIM00780
C ISEED = INITIALIZATION FOR RANDOM NUMBER GENERATOR SIM00790
C XKAL = X MATRIX SIM00800
C Y = OUTPUT MATRIX SIM00810
C RNDM = RANDOM NUMBERS USED FOR WHITE NOISE IN MEASUREMENTS AND SIM00820
C IN PLANT FORCES SIM00830
C BN = B MATRIX TO MULTIPLY NOISE DISTURBANCES SIM00840
C TNX,TNY,TNZ= NOISE TORQUES X,Y,Z SLOPES SIM00850
C M2=2*MODE SIM00860
C
C *****
C ***** SAMPLE OF SPACE EXEC FILE *****
C
C THIS FILE MUST BEGIN IN COLUMN 1 AND RUN WITH THE FOLLOWING SIM00890
C SEQUENCE FOR THE INITIAL RUN OF THE PROGRAM: SIM00900
C
C FORTVS SPACE (COMPILES PROGRAM) SIM00950
C SPACE (EXECUTES EXEC FILE) SIM00960
C LOAD SPACE (START) (LOADS AND EXECUTES PROGRAM) SIM00970
C
C SUBSEQUENT PROGRAM RUNS CAN ELIMINATE "FORTVS SPACE" IF NO SIM00980
C CHANGES HAVE BEEN MADE TO THE PROGRAM, AND CAN ELIMINATE SIM00990
C RUNNING THE EXEC FILE. SIM01000
C
C FI 4 DISK THESIS INPUT B (PERM SIM01010
C FI 8 DISK UTILITY DATA (RECFM VS BLOCK 133 PERM SIM01020
C FI 11 DISK CNTRL OUTPUT (RECFM F BLOCK 80 LRECL 80 PERM SIM01030
C FI 13 DISK GAMMA OUTPUT (RECFM VS BLOCK 133 PERM SIM01040
C FI 14 DISK MODE OUTPUT (RECFM F BLOCK 80 LRECL 80 PERM SIM01050
C

```

```

C   FI 16 DISK COST OUTPUT  (RECFM F BLOCK 80 LRECL 80 PERM SIM01080
C   FI 17 DISK PRT  OUTPUT  (RECFM F BLOCK 80 LRECL 80 PERM SIM01090
C   FI 18 DISK ERROR DATA  (RECFM F BLOCK 80 LRECL 80 PERM SIM01100
C   FI 19 DISK END FILE   (RECFM F BLOCK 80 LRECL 80 PERM SIM01110
C   FI 20 DISK GMAT FILE  (RECFM F BLOCK 80 LRECL 80 PERM SIM01120
C
C   ****
C   PARAMETER (JR=5243, KR=5397, MR=262139) SIM01130
C
C   MIN =1.00 SIM01140
C
C   WT=1.0D00 SIM01150
C   PI = 4.0D0 * ATAN(1.0D0) SIM01160
C
C   ****
C   ***** READ LAMA AND UGVEX MATRICIES ***** SIM01170
C   ***** SIM01180
C
C   CALL EXCMS ('CLRSCRN') SIM01190
C   WRITE(6,1008) SIM01200
C   WRITE(6,*) ' READING LAMA AND UGVEX MATRICIES' SIM01210
C   WRITE(6,*) ' SIM01220
C
C   THIS SECTION READS THE LAMA VECTOR AND THE UGVEX SIM01230
C   MATRIX AND STORES THEM IN MEMORY FOR FURTHER RECALL OF SIM01240
C   DESIRED LOCATION DATA. SIM01250
C
C   READ(4,1001) NAM SIM01260
C   READ(4,1002)(LAMA(I),I=1,100) SIM01270
C   READ(4,1001) NAM SIM01280
C   DO 5 J = 1,100 SIM01290
C     READ(4,1002)(UGVEX(I,J),I=1,684) SIM01300
C
5   CONTINUE SIM01310
C
1001 FORMAT(1X,A6) SIM01320
1002 FORMAT(1X,8E15.8) SIM01330
1008 FORMAT (1X,///) SIM01340
C
500 CALL EXCMS ('CLRSCRN') SIM01350
C
C   **** STARTING MODE NUMBER **** SIM01360
C   ** SMODE 7 TO 100 (INTEGER) ** SIM01370
C   SMODE= 7 SIM01380
C
C   WRITE (16,700) SMODE SIM01390
700 FORMAT (' ','STARTING MODE NUMBER: ',I2) SIM01400
C
C   **** NUMBER OF MODES TO SCAN **** SIM01410
C   ** MODE 1 TO 93 (INTEGER) SIM01420
C
C   MODE=20 SIM01430
C
C   EMODE = SMODE + MODE - 1 SIM01440
C
C

```

```

      WRITE (16,701) MODE          SIM01630
701  FORMAT (' ','NUMBER OF MODES SCANNED: ',I2)  SIM01640
C
C  ***** NOISE INPUT POSITION      *****
C  ** NODE 1 TO 114 (INTEGER) (IF 0 THEN NO NOISE INPUT)  SIM01650
C  NODE= 8          SIM01660
C
C  WRITE (16,702) NODE          SIM01670
702  FORMAT (' ','NOISE NODE LOCATION: ',I5)  SIM01680
C
C  ***** START AND STOP FOR PLANT  SIM01690
SN=7          SIM01700
FN=20          SIM01710
NS=SN*2-1      SIM01720
NF=SN*2+2*FN-2  SIM01730
WRITE (16,899) SN,FN          SIM01740
899  FORMAT (' ','PLANT -- SN= ',I5,' FN= ',I5)  SIM01750
C  ***** SAMPLING TIME          *****  SIM01760
C  ** SAMPT MUST BE LESS THAN OR EQUAL TO SAMPTM **  SIM01770
C  SAMPT = 0.05          SIM01780
SAMPTM = ((2.0D0*PI)/SQRT(LAMA(EMODE)))/1.0D01  SIM01790
IF (SAMPT.GE.SAMPTM) THEN  SIM01800
  SAMPT=SAMPTM  SIM01810
ENDIF  SIM01820
C
C  WRITE (16,900) MIN          SIM01830
900  FORMAT (' ',2X,'MIN: ',F8.3)  SIM01840
C
C  WRITE (16,703) SAMPT, SAMPTM  SIM01850
703  FORMAT (' ','SAMPLING TIME: ',D12.4,2X,'SAMPTM= ',D15.8)  SIM01860
C
C  ***** DAMPING FACTOR          *****
C  ** DAMP 0.0 TO 1.0 (REAL*8)  SIM01870
C  DAMP=.01          SIM01880
C
C  WRITE (16,704) DAMP          SIM01890
704  FORMAT (' ','DAMPING FACTOR: ',D12.4)  SIM01900
C
C
C  *** PLANT NOISE VARIANCE ***
C  ** PNVARX, PNVARY, PNVARZ GT 0.0  SIM01910
SF1=2.5D06  SIM01920
SF=1.0D00  SIM01930
C
C  PNVARX=1.0D00*SF1  SIM01940
C  PNVARY=1.0D00*SF1  SIM01950
C  PNVARZ=1.0D00*SF1  SIM01960
C
C
C  *** MEASUREMENT NOISE VARIANCE ***
C  ** MNVARX, MNVARY, MNVARZ GT 0.0  SIM01970
MNVARX=1.0D-03 *SF  SIM01980
MNVARY=1.0D-03 *SF  SIM01990

```

```

MNVARZ=1.0D-03 *SF SIM02170
C SIM02180
C SIM02190
C SIM02200
C SIM02210
C SIM02220
C SIM02230
C SIM02240
C SIM02250
C SIM02260
C SIM02270
C SIM02280
C SIM02290
C SIM02300
C SIM02310
C SIM02320
C SIM02330
C SIM02340
C SIM02350
C ***** NOISE INPUT LOCATION ***** SIM02360
C SIM02370
C ROWN3 = NODE*6 SIM02380
C ROWN2 = (NODE*6) - 1 SIM02390
C ROWN1 = (NODE*6) - 2 SIM02400
C COUNT = 0 SIM02410
C SIM02420
C SIM02430
C ***** INITIALIZE MATRICIES ***** SIM02440
C SIM02450
C DO 48 K=1,50 SIM02460
C IDENT(K,K)=1.0 SIM02470
48 CONTINUE SIM02480
C SIM02490
C DO 54 K = 1, 200 SIM02500
C X(K) = 0.0 SIM02510
54 CONTINUE SIM02520
C SIM02530
C SIM02540
C WRITE(6,1008) SIM02550
C WRITE (6,*) ' INITIALIZE RMN AND QPN MATRICES ' SIM02560
C *** INITIALIZE RMN AND QPN MATRICES *** SIM02570
C SIM02580
C DO 60 I=1,3 SIM02590
C DO 58 J=1,3 SIM02600
C RMN(I,J)=0.0 SIM02610
C QPN(I,J)=0.0 SIM02620
58 CONTINUE SIM02630
60 CONTINUE SIM02640
C SIM02650
C RMN(1,1)=MNVARX**2 SIM02660
C RMN(2,2)=MNVARY**2 SIM02670
C RMN(3,3)=MNVARZ**2 SIM02680
C QPN(1,1)=PNVARX**2 SIM02690
C QPN(2,2)=PNVARY**2 SIM02700

```

```

QPN(3,3)=PNVARZ***2.0 SIM02710
C SIM02720
C SIM02730
C WRITE(6,1008) SIM02740
C ' ENTER STMTRX SIM02750
C ***** SIM02760
C BEGIN MAIN PROGRAM ***** SIM02770
C ***** SIM02780
C ***** SIM02790
C CALL STMTRX(EMODE,SMODE,SAMPT,DAMP,PHI,GAMMA,A,B,LAMA,UGVEX,C, SIM02800
+ ROWN1,ROWN2,ROWN3,BN) SIM02810
C SIM02820
C SIM02830
C WRITE (16,1008) SIM02840
DO 11000 I=1,200 SIM02850
    DO 10900 J=1,3 SIM02860
        C(J,I)= C( J,I)*SF SIM02870
10900    CONTINUE SIM02880
11000    CONTINUE SIM02890
C SIM02900
C ***** PRE-LOOP PORTION OF KALMAN FILTER SIM02910
C SIM02920
C M2=2*MODE SIM02930
JP=2*SMODE-1 SIM02940
JQ=2*EMODE SIM02950
DO 90 I=1,50 SIM02960
    XH(I)=0.0 SIM02970
90    CONTINUE SIM02980
C SIM02990
C DO 9971 I=1,M2 SIM03000
    READ (20,*) G(I,1), G(I,2), G(I,3) SIM03010
9971    CONTINUE SIM03020
C SIM03030
C WRITE (14,1008) SIM03040
DO 384 I=1,M2 SIM03050
    WRITE (14,5350) G(I,1),G(I,2),G(I,3) SIM03060
384    CONTINUE SIM03070
5350    FORMAT (' ',2X,D15.8,2X,D15.8,2X,D15.8) SIM03080
C SIM03090
C SIM03100
C ***** THIS SECTION COMPUTES THE STATE UPDATE ***** SIM03110
C ***** SIM03120
C DO 9771 I=1,100 SIM03130
    E2(I)=0.0 SIM03140
    E3(I)=0.0 SIM03150
    XS(I)=0.0 SIM03160
9771    CONTINUE SIM03170
    ESS =0.0 SIM03180
    COUNT = 0 SIM03190
    ENERGY = 0.0D0 SIM03200
    TIME = 0.0 SIM03210
    CGN=0.0 SIM03220
SIM03230
SIM03240
SIM03250

```

```

CTG=0.0
C      **** SETS LOOP FOR THE ITERATIONS NECESSARY TO OBSERVE      ****
C      **** THE SYSTEM FOR THE NUMBER OF MINUTES SPECIFIED      ****
C      WRITE (6,1008)
C      WRITE (6,*)          START STATE UPDATE      '
C      LOOP = INT((MIN*60.0)/SAMPT)
C      PRT= (DBLE(LOOP))/30.0
C      CTT=0.0
C
C      DO 400 L = 0, LOOP
C          TIME = DBLE(L)*SAMPT
C
C      IF(L.EQ.0)THEN
C          IMPLSE =(1.0D06*SF1)/(DSQRT(SAMPT))
C      ELSE
C          IMPLSE = 0.0D0
C      ENDIF
C
C      TO=0.0
C      **** RANDOM NUMBER GENERATOR ****
C
C      DO 101 I=1,6
C          ISEED=MOD(ISEED*JR+KR,MR)
C          RND1=(DBLE(ISEED)+0.5D00)/DBLE(MR)
C          ISEED=MOD(ISEED*JR+KR,MR)
C          RND2=(DBLE(ISEED)+0.5D00)/DBLE(MR)
C          RNDM(I)=DSQRT(-2.0*DLOG(RND1))*DCOS(6.2831853D00*RND2)
101    CONTINUE
C      ****
C      CTT=CTT+1.0
C      **** START OF STATE UPDATE ***
C
C      **** COMPUTE AX0200 = A0200 X 200 * X0200
C
C      **** COMPUTE AX = A0X
C
C      JK=SMODE*2-2
C      JP=JK+1
C      JQ=2*EMODE
C
C      DO 5015 I=NS,NF
C          SUM=0.0
C              DO 5010 K=NS,NF
C                  SUM=SUM+A(I,K)*X(K)
5010    CONTINUE
C                  AX(I)=SUM
5015    CONTINUE
C
C      **** COMPUTE WD03
C
C      WD(1)=PNVARX*RNDM(1)*TO+IMPLSE
C      WD(2)=PNVARY*RNDM(2)*TO
C      WD(3)=PNVARZ*RNDM(3)*TO
C

```

```

C          SIM03820
C          *** COMPUTE BNWD0200 =BN0200 X 3 * WD03
C          SIM03830
C          SIM03840
C          DO 5035 I=NS,NF
C          SUM=0.0
C          DO 5030 K=1,3
C          SUM=SUM+BN(I,K)*WD(K)
C          CONTINUE
C          BNWD(I)=SUM
C          CONTINUE
C          SIM03850
C          SIM03860
C          SIM03870
C          SIM03880
C          SIM03890
C          SIM03900
C          SIM03910
C          SIM03920
C          SIM03930
C          SIM03940
C          SIM03950
C          SIM03960
C          SIM03970
C          SIM03980
C          SIM03990
C          SIM04000
C          SIM04010
C          SIM04020
C          SIM04030
C          SIM04040
C          SIM04050
C          SIM04060
C          SIM04070
C          SIM04080
C          SIM04090
C          SIM04100
C          SIM04110
C          SIM04120
C          SIM04130
C          SIM04140
C          SIM04150
C          SIM04160
C          SIM04170
C          SIM04180
C          SIM04190
C          SIM04200
C          SIM04210
C          SIM04220
C          *** START OF KALMAN FILTER ***
C          SIM04230
C          SIM04240
C          M2=2*MODE
C          SIM04250
C          SIM04260
C          SIM04270
C          SIM04280
C          SIM04290
C          SIM04300
C          SIM04310
C          SIM04320
C          SIM04330
C          SIM04340
C          SIM04350
C          SIM04360
C          SIM04370
5030
5035
C          DO 5040 I=NS,NF
C          X(I)= AX(I) + BNWD(I)
C          IF (DABS(X(I)).LT. 1.0D-60) THEN
C              X(I)=1.0D-60
C          END IF
C
C
C
5040
C          COMPUTE V03
C
C          V(1)=MNVARX*RNDM(4)
C          V(2)=MNVARY*RNDM(5)
C          V(3)=MNVARZ*RNDM(6)
C
C          *** COMPUTE Y03 = C03 X 200 * X0200 + V03
C
C          DO 5050 I=1,3
C          SUM=0.0
C          DO 5045 K=NS,NF
C          SUM=SUM+C(I,K)*X(K)
C          CONTINUE
C          Y(I)=SUM+V(I)
5045
5050
C          ****
C
C          **** START OF KALMAN FILTER ***
C
C          M2=2*MODE
C
C          *** COMPUTE XH1 = A*XH
C
C          DO 300 I=JP,JQ
C          SUM=0.0
C          DO 295 K=JP,JQ
C          SUM=SUM+A(I,K) * XH(K)
C          CONTINUE
C          XH1(I)=SUM
295
300
C          ****

```

```

C          *** COMPUTE DY = Y - C*XH1          SIM04380
C          DO 315 I=1,3                      SIM04390
C          SUM=0.0                           SIM04400
C          DO 310 K=JP,JQ                   SIM04410
C          SUM=SUM+C(I,K)*XH1(K)           SIM04420
310          CONTINUE                         SIM04430
C          DY(I)=Y(I)-SUM                  SIM04440
315          CONTINUE                         SIM04450
C          ****
C          *** COMPUTE XH = XH1 + G*DY          SIM04460
C          DO 325 I=1,M2                   SIM04470
C          J1=JP-1+I                      SIM04480
C          SUM=0.0                           SIM04490
C          DO 320 K=1,3                   SIM04500
C          SUM=SUM+G(I,K)*DY(K)           SIM04510
320          CONTINUE                         SIM04520
C          XH(J1)=XH1(J1)+SUM             SIM04530
C          IF (DABS(XH(J1)). LT. 1.0D-60) THEN SIM04540
C          XH(J1)=1.0*D-60                SIM04550
C          END IF                           SIM04560
C          CONTINUE                         SIM04570
325          CONTINUE                         SIM04580
C          **** END OF KALMAN ROUTINES ****      SIM04590
C          *** COMPUTATION OF ESUM ***
DO 340 I=JP,JQ                      SIM04600
XDEL= X(I)-XH(I)                      SIM04610
XDEL1=XDEL*XDEL*SAMPT                 SIM04620
E2(I)=E2(I)+XDEL1                     SIM04630
XS(I)=XS(I)+X(I)*X(I)*SAMPT          SIM04640
E3(I)=E2(I)/XS(I)                     SIM04650
340          CONTINUE                         SIM04660
C          CGN=CGN+1.0                      SIM04670
C          IF (CTT.EQ.1.0.OR.CGN.GT.PRT) THEN SIM04680
C          WRITE (6,*) 'TIME= ', TIME, ' SEC.' SIM04690
C          WRITE (17,1008)                   SIM04700
C          WRITE (16,1008)                   SIM04710
C          WRITE (16,2100) TIME              SIM04720
C          WRITE (17,2100) TIME              SIM04730
2100          FORMAT(' ', 'TIME= ', F9.3)  SIM04740
          DO 380 I=JP, JQ                 SIM04750
          WRITE (16,4500) I,X(I) ,I ,XH(I)  SIM04760
C          .
C          WRITE (17,2100) TIME              SIM04770
          FORMAT(' ', 'TIME= ', F9.3)  SIM04780
          DO 380 I=JP, JQ                 SIM04790
          WRITE (16,4500) I,X(I) ,I ,XH(I)  SIM04800
C          .
C          WRITE (17,2100) TIME              SIM04810
          FORMAT(' ', 'TIME= ', F9.3)  SIM04820
          DO 380 I=JP, JQ                 SIM04830
          WRITE (16,4500) I,X(I) ,I ,XH(I)  SIM04840
C          .
C          WRITE (17,2100) TIME              SIM04850
          FORMAT(' ', 'TIME= ', F9.3)  SIM04860
          DO 380 I=JP, JQ                 SIM04870
          WRITE (16,4500) I,X(I) ,I ,XH(I)  SIM04880
C          .
C          WRITE (17,2100) TIME              SIM04890
          FORMAT(' ', 'TIME= ', F9.3)  SIM04900
          DO 380 I=JP, JQ                 SIM04910
          WRITE (16,4500) I,X(I) ,I ,XH(I)  SIM04920

```

```

      WRITE (17,4530) I,E2(I) ,E3(I) , XS(I)          SIM04930
380  CONTINUE                                     SIM04940
C
C
C
C
      CGN=0.0                                     SIM04950
      END IF                                     SIM04960
4500  FORMAT (', ', ' X( ',I3, ')= ',D15.8,2X,'XH( ',I3, ')= ',D15.8)  SIM05010
4530  FORMAT (' ',5X,I5,5X,3 D15.8)           SIM05020
C
400   CONTINUE                                     SIM05030
C
C
      DO 401 I=JP,JQ
      WRITE (19,4530) I, E2(I) ,E3(I), XS(I)          SIM05040
401   CONTINUE                                     SIM05050
C
C
C
C
599   STOP                                         SIM05060
      END                                           SIM05070
C
C
C
C
      *****
C
C
      THIS SUBROUTINE COMPUTES THE STATE TRANSITION MATRIX FOR EACH
C
      OF THE 100 MODES
      *****
C
C
      SUBROUTINE STMTRX(EMODE,SMODE,T,D,PHI,GAMMA,A,B,LAMA,UGVEX,C,
+                      ROWN1,ROWN2,ROWN3,BN)          SIM05130
C
      REAL*8 WN,GMA,PHI(2,2,100),GAMMA(2,100),EGT,T,COSW1T,SINW1T  SIM05140
      REAL*8 W1,D,A(200,200),B(200,3),C(3,200),BN(200,3)          SIM05150
      REAL LAMA(100),UGVEX(684,100)          SIM05160
      INTEGER SMODE,R,EMODE,JJ,KK,ROWN1,ROWN2,ROWN3          SIM05170
C
C
      DO 600 I = 1      ,100
      WN = DBLE(SQRT(LAMA(I)))          SIM05230
      GMA = D*WN/2.0          SIM05240
      EGT = DEXP(-GMA*T)          SIM05250
      W1 = DSQRT((WN**2)-(GMA**2))          SIM05260
      COSW1T = DCOS(W1*T)          SIM05270
      SINW1T = DSIN(W1*T)          SIM05280
C
C
      DO 600 I = 1      ,100
      WN = DBLE(SQRT(LAMA(I)))          SIM05290
      GMA = D*WN/2.0          SIM05300
      EGT = DEXP(-GMA*T)          SIM05310
      W1 = DSQRT((WN**2)-(GMA**2))          SIM05320
      COSW1T = DCOS(W1*T)          SIM05330
      SINW1T = DSIN(W1*T)          SIM05340
C
C
      IF(WN.EQ.0)THEN          SIM05350
          PHI(1,1,I) = EGT*COSW1T          SIM05360
          PHI(1,2,I) = T          SIM05370
          PHI(2,1,I) = 0          SIM05380
          PHI(2,2,I) = EGT*COSW1T          SIM05390
C
C
      IF(WN.EQ.0)THEN          SIM05400
          PHI(1,1,I) = EGT*COSW1T          SIM05410
          PHI(1,2,I) = T          SIM05420
          PHI(2,1,I) = 0          SIM05430
          PHI(2,2,I) = EGT*COSW1T          SIM05440
C
C
      IF(WN.EQ.0)THEN          SIM05450
          PHI(1,1,I) = EGT*COSW1T          SIM05460
          PHI(1,2,I) = T          SIM05470
          PHI(2,1,I) = 0          SIM05480
          PHI(2,2,I) = EGT*COSW1T

```

```

C SIM05490
C SIM05500
C SIM05510
C SIM05520
C SIM05530
C SIM05540
C SIM05550
C SIM05560
C SIM05570
C SIM05580
C SIM05590
C SIM05600
C SIM05610
C SIM05620
C SIM05630
C SIM05640
C SIM05650
C SIM05660
C SIM05670
C SIM05680
C SIM05690
C SIM05700
C SIM05710
C SIM05720
C SIM05730
C SIM05740
C SIM05750
C SIM05760
C SIM05770
C SIM05780
C SIM05790
C SIM05800
C SIM05810
C SIM05820
C SIM05830
C SIM05840
C SIM05850
C SIM05860
C SIM05870
C SIM05880
C SIM05890
C SIM05900
C SIM05910
C SIM05920
C SIM05930
C SIM05940
C SIM05950
C SIM05960
C SIM05970
C SIM05980
C SIM05990
C SIM06000
C SIM06010
C SIM06020
C SIM06030
C SIM06040

      GAMMA(1,I) = 0
      GAMMA(2,I) = 0
      ELSE
C
C
C
C
      PHI(1,1,I) = EGT*(COSW1T + (GMA*(W1**(-1)))*SINW1T)
      PHI(1,2,I) = (W1**(-1))*EGT*SINW1T
      PHI(2,1,I) = -(WN**2)*(W1**(-1))*EGT*SINW1T
      PHI(2,2,I) = EGT*(COSW1T - (GMA*(W1**(-1)))*SINW1T)
C
C
C
C
      GAMMA(1,I)=(WN**(-2))*(1.0D0-EGT*COSW1T -EGT*(GMA/W1)*SINW1T)
      GAMMA(2,I) = (W1**(-1))*EGT*SINW1T
C
C
C
C
      ENDIF
C
C
C
600    CONTINUE
C
C
      R = 1
C
      DO 610 K = 1      ,100
C
C
C
C
      A(R,R) = PHI(1,1,K)
      A(R,R+1) = PHI(1,2,K)
      A(R+1,R) = PHI(2,1,K)
      A(R+1,R+1) = PHI(2,2,K)
C
C
C
C
      **** B MATRIX FOR MULTIPLYING CONTROL TORQUES
C
      B(R,1) = GAMMA(1,K)*DBLE(UGVEX(412,K))
      B(R,2) = GAMMA(1,K)*DBLE(UGVEX(413,K))
      B(R,3) = GAMMA(1,K)*DBLE(UGVEX(414,K))
      B(R+1,1) = GAMMA(2,K)*DBLE(UGVEX(412,K))

```


C
C

RETURN
END

SIM06610
SIM06620
SIM06630
SIM06640

APPENDIX C. PROGRAM TO ESTIMATE NOISE IN KALMAN FILTER FROM UNOBSERVED MODES

```

***** SPAC 24 *****
***** ADAPTED TO RUN N MODES OF THE PLANT AND *****
***** COMPUTE THE NOISE IN THE KALMAN FILTER *****
***** FROM THE UNOBSERVED MODES *****
***** VARIABLE DECLARATIONS *****

EXTERNAL STMTRX,EXCMS
CHARACTER*6 NAM
CHARACTER*1 AGAIN, CORECT, RAGAIN
INTEGER ROWN1, ROWN2, ROWN3, COUNT, NODE, MODE, KQ, EMODE, SMODE, R2M, C2M
INTEGER CT, CF, KADJ, CFADJ, LOOP, PRNT, JJ, JK, N1, JR, KR, MR, ISEED, M2
INTEGER NO, NS, NF, SN, FN
INTEGER JL, J1, JM, JP, JQ, KA, KB, KC, KD, KE, KF, KG

REAL LAMA( 100 ), UGVEX( 684, 100 ), RNODE, RMODE, MIN
REAL*8 PHI( 2, 2, 100 ), GAMMA( 2, 100 ), EGT, GMA, WN, W1, X1T, X2T, TIME
REAL*8 A( 200, 200 ), B( 200, 3 ), F( 3, 50 ), IMPLSE, ENERGY
REAL*8 COSW1T, SINW1T, X( 200 )
REAL*8 DAMP, SAMPT, PI, SAMPTM, SUM1, SUM2, SUM3, SUMC
REAL*8 C( 9, 200 ), RMN( 3, 3 ), QPN( 3, 3 )
REAL*8 BN( 200, 3 )
REAL*8 PNVARX, PNVARY, PNVARZ
REAL*8 MNVARX, MNVARY, MNVARZ
REAL*8 SUM, RNDM( 6 ), RND1, RND2
REAL*8 ES, ED, ESUM, CGN, PRT
REAL*8 WT, WD( 3 ), BNWD( 200 ), EX1( 9 )
REAL*8 EX( 9 ), AX( 200 ), SF, TO, CTT, ESS
REAL*8 CTG, XDEL, XDEL1, ERS, PRT1
REAL*8 SF1

***** VARIABLE DEFINITIONS *****

STMTRX = SUBROUTINE ESTABLISHES STATE TRANSITION MATRICES
LAMA = VECTOR OF THE SQUARE OF THE NATURAL FREQUENCIES
UGVEX = MODE POSITIONS AND SLOPES OF THE NODAL POINTS
PHI = STATE TRANSITION MATRICES FOR EACH MODE
GAMMA = INPUT TRANSITION MATRIX
A = DIAGONAL MATRIX CONSISTING OF PHI

```

C	B = INPUT MATRIX OF GAMMA AND CONTROL SLOPES	SPA00500
C	DAMP = DAMPING FACTOR	SPA00510
C	SAMPT = SAMPLING TIME	SPA00520
C	IMPLSE = IMPULSE INPUT FUNCTION	SPA00530
C	MIN = NUMBER OF MINUTES SYSTEM WILL BE OBSERVED	SPA00540
C	SMODE = NUMBER OF STARTING MODE (INT)	SPA00550
C	MODE = NUMBER OF MODES (INT)	SPA00560
C	EMODE = NUMBER OF THE LAST MODE (INT)	SPA00570
C	NODE = NUMBER OF THE NOISE INPUT MODE (INT)	SPA00580
C	*** NOISE SLOPE LOCATIONS IN DATA MATRIX ***	SPA00590
C	ROWN1 = X-SLOPE LOCATION	SPA00600
C	ROWN2 = Y-SLOPE LOCATION	SPA00610
C	ROWN3 = Z-SLOPE LOCATION	SPA00620
C	C = OUTPUT MATRIX FOR Y	SPA00630
C	IDENT = IDENTITY MATRIX	SPA00640
C	RMN = MEASUREMENT NOISE COVARIANCE MATRIX	SPA00650
C	QPN = PLANT NOISE COVARIANCE MATRIX	SPA00660
C	PNVARX = PLANT NOISE X-SLOPE VARIANCE	SPA00670
C	PNVARY = PLANT NOISE Y-SLOPE VARIANCE	SPA00680
C	PNVARZ = PLANT NOISE Z-SLOPE VARIANCE	SPA00690
C	MNVARX = MEASUREMENT NOISE X-SLOPE VARIANCE	SPA00700
C	MNVARY = MEASUREMENT NOISE Y-SLOPE VARIANCE	SPA00710
C	MNVARZ = MEASUREMENT NOISE Z-SLOPE VARIANCE	SPA00720
C	ISEED = INITIALIZATION FOR RANDOM NUMBER GENERATOR	SPA00730
C	RNDM = RANDOM NUMBERS USED FOR WHITE NOISE IN MEASUREMENTS AND IN PLANT FORCES	SPA00740
C	BN = B MATRIX TO MULTIPLY NOISE DISTURBANCES	SPA00750
C		SPA00760
C		SPA00770
C		SPA00780
C		SPA00790

***** SAMPLE OF SPACE EXEC FILE *****

THIS FILE MUST BEGIN IN COLUMN 1 AND RUN WITH THE FOLLOWING
SEQUENCE FOR THE INITIAL RUN OF THE PROGRAM:

C	FORTVS SPACE	(COMPILES PROGRAM)	SPA00850
C	SPACE	(EXECUTES EXEC FILE)	SPA00860
C	LOAD SPACE (START)	(LOADS AND EXECUTES PROGRAM)	SPA00870

SUBSEQUENT PROGRAM RUNS CAN ELIMINATE "FORTVS SPACE" IF NO
CHANGES HAVE BEEN MADE TO THE PROGRAM, AND CAN ELIMINATE
RUNNING THE EXEC FILE.

C	FI 4 DISK THESIS INPUT (PERM	SPA00930
C	FI 8 DISK UTILITY DATA (RECFM VS BLOCK 133 PERM	SPA00940
C	FI 11 DISK CNTRL OUTPUT (RECFM F BLOCK 80 LRECL 80 PERM	SPA00950
C	FI 13 DISK GAMMA OUTPUT (RECFM VS BLOCK 133 PERM	SPA00960
C	FI 14 DISK MODE OUTPUT (RECFM F BLOCK 80 LRECL 80 PERM	SPA00970
C	FI 16 DISK COST OUTPUT (RECFM F BLOCK 80 LRECL 80 PERM	SPA00980
C	FI 17 DISK PRT OUTPUT (RECFM F BLOCK 80 LRECL 80 PERM	SPA00990
C	FI 18 DISK ERROR DATA (RECFM F BLOCK 80 LRECL 80 PERM	SPA01000
C	FI 19 DISK END FILE (RECFM F BLOCK 80 LRECL 80 PERM	SPA01010
C	FI 20 DISK GMAT FILE (RECFM F BLOCK 80 LRECL 80 PERM	SPA01020

C	SPA01030
C	SPA01040
C	SPA01050

```

PARAMETER (JR=5243, KR=5397, MR=262139) SPA01060
C SPA01070
C SPA01080
C MIN = 15.0 SPA01090
C SPA01100
C WT=1.0D00 SPA01110
C PI = 4.0D0 * ATAN(1.0D0) SPA01120
C ***** SPA01130
C READ LAMA AND UGVEX MATRICIES ***** SPA01140
C ***** SPA01150
C ***** SPA01160
C ***** SPA01170
C CALL EXCMS ('CLRSCRN') SPA01180
C WRITE(6,1008) SPA01190
C WRITE(6,*)' READING LAMA AND UGVEX MATRICIES' SPA01200
C WRITE(6,*)' SPA01210
C THIS SECTION READS THE LAMA VECTOR AND THE UGVEX SPA01220
C MATRIX AND STORES THEM IN MEMORY FOR FURTHER RECALL OF SPA01230
C DESIRED LOCATION DATA. SPA01240
C SPA01250
C READ(4,1001) NAM SPA01260
C READ(4,1002)(LAMA(I),I=1,100) SPA01270
C READ(4,1001) NAM SPA01280
C DO 5 J = 1,100 SPA01290
C READ(4,1002)(UGVEX(I,J),I=1,684) SPA01300
5 CONTINUE SPA01310
C SPA01320
1001 FORMAT(1X,A6) SPA01330
1002 FORMAT(1X,8E15.8) SPA01340
1008 FORMAT(1X,///) SPA01350
C SPA01360
500 CALL EXCMS ('CLRSCRN') SPA01370
C SPA01380
C ***** STARTING MODE NUMBER ***** SPA01390
C *** SMODE 7 TO 100 (INTEGER) *** SPA01400
C SMODE= 17 SPA01410
C SPA01420
C WRITE (16,700) SMODE SPA01430
700 FORMAT (' ','STARTING MODE NUMBER: ',I2) SPA01440
C SPA01450
C ***** NUMBER OF MODES TO SCAN ***** SPA01460
C ** MODE 1 TO 93 (INTEGER) SPA01470
C SPA01480
C MODE=3 SPA01490
C SPA01500
C EMODE = SMODE + MODE - 1 SPA01510
C SPA01520
C WRITE (16,701) MODE SPA01530
701 FORMAT (' ','NUMBER OF MODES SCANNED: ',I2) SPA01540
C SPA01550
C ***** NOISE INPUT POSITION ***** SPA01560
C ** NODE 1 TO 114 (INTEGER) (IF 0 THEN NO NOISE INPUT) SPA01570
C NODE= 8 SPA01580
C SPA01590
C WRITE (16,702) NODE SPA01600
702 FORMAT (' ','NOISE NODE LOCATION: ',I5) SPA01610

```

```

C ***** START AND STOP FOR PLANT SPA01620
C SN=17 SPA01630
C FN=4 SPA01640
C NS=SN*2-1 SPA01650
C NF=SN*2+2*FN-2 SPA01660
C WRITE (16,899) SN,FN SPA01670
899 FORMAT (' ', 'PLANT -- SN= ',I5,' FN= ',I5) SPA01680
C ***** SAMPLING TIME ***** SPA01690
C *** SAMPT MUST BE LESS THAN OR EQUAL TO SAMPTM *** SPA01700
C SAMPT = 0.05 SPA01710
C SAMPTM = ((2.0D0*PI)/SQRT(LAMA(EMODE)))/1.0D01 SPA01720
C IF (SAMPT.GE.SAMPTM) THEN SPA01730
C SAMPT=SAMPTM SPA01740
C ENDIF SPA01750
C
C WRITE (16,900) MIN SPA01760
900 FORMAT (' ',2X,'MIN: ',F8.3) SPA01770
C
C WRITE (16,703) SAMPT, SAMPTM SPA01780
703 FORMAT (' ','SAMPLING TIME: ',D12.4,2X,'SAMPTM= ',D15.8) SPA01790
C ***** DAMPING FACTOR ***** SPA01800
C *** DAMP 0.0 TO 1.0 (REAL*8) SPA01810
C DAMP=.01 SPA01820
C
C WRITE (16,704) DAMP SPA01830
704 FORMAT (' ','DAMPING FACTOR: ',D12.4) SPA01840
C
C NO=3 SPA01850
C *** PLANT NOISE VARIANCE *** SPA01860
C ** PNVARX, PNVARY, PNVARZ GT 0.0 SPA01870
C
C SF=1.0D0 SPA01880
C SF1=2.5D06 SPA01890
C PNVARX=1.0D00*SF1 SPA01900
C PNVARY=1.0D00*SF1 SPA01910
C PNVARZ=1.0D00*SF1 SPA01920
C
C *** MEASUREMENT NOISE VARIANCE *** SPA01930
C ** MNVARX, MNVARY, MNVARZ GT 0.0 SPA01940
C MNVARX=1.0D-03 *SF SPA01950
C MNVARY=1.0D-03 *SF SPA01960
C MNVARZ=1.0D-03 *SF SPA01970
C
C WRITE (16,711) SPA01980
711 FORMAT(' ','PLANT NOISE VARIANCE: ') SPA01990
C WRITE (16,712) SPA02000
712 FORMAT(' ',6X,'PNVARX',13X,'PNVARY',13X,'PNVARZ') SPA02010
C WRITE (16,713) PNVARX, PNVARY, PNVARZ SPA02020
713 FORMAT(' ',2X,E15.8,2X,E15.8,2X,E15.8) SPA02030
C WRITE(16,714) SPA02040
714 FORMAT(' ','MEASUREMENT NOISE: ') SPA02050

```

```

    WRITE(16,715) SPA02180
715  FORMAT(' ',6X,'MNVARX',13X,'MNVARY',13X,'MNVARZ') SPA02190
      WRITE(16,713) MNVARX,MNVARY,MNVARZ SPA02200
C
510  CALL EXCMS ('CLRSCRN') SPA02210
      WRITE (6,1008) SPA02220
      WRITE (6,*) ' PROGRAM RUNNING' SPA02230
C
C      ***** NOISE INPUT LOCATION ***** SPA02240
C
      ROWN3 = NODE*6 SPA02250
      ROWN2 = (NODE*6) - 1 SPA02260
      ROWN1 = (NODE*6) - 2 SPA02270
      COUNT = 0 SPA02280
C
C      ***** INITIALIZE MATRICIES ***** SPA02290
C
      DO 54 K = 1, 200 SPA02300
          X(K) = 0.0 SPA02310
54    CONTINUE SPA02320
C
      DO 60 I=1,3 SPA02330
          DO 58 J=1,3 SPA02340
              RMN(I,J)=0.0 SPA02350
              QPN(I,J)=0.0 SPA02360
58    CONTINUE SPA02370
60    CONTINUE SPA02380
C
      RMN(1,1)=MNVARX**2.0 SPA02390
      RMN(2,2)=MNVARY**2.0 SPA02400
      RMN(3,3)=MNVARZ**2.0 SPA02410
      QPN(1,1)=PNVARX**2.0 SPA02420
      QPN(2,2)=PNVARY**2.0 SPA02430
      QPN(3,3)=PNVARZ**2.0 SPA02440
C
C      ***** BEGIN MAIN PROGRAM ***** SPA02450
C
      CALL STMTRX(EMODE,SMODE,SAMPT,DAMP,PHI,GAMMA,A,B,LAMA,UGVEX,C,
+      ROWN1,ROWN2,ROWN3,BN) SPA02460
C
C
      WRITE (6,1008) SPA02470
      WRITE(6,*) ' EXIT STMTRX - - - PRE-LOOP KALMAN' SPA02480
C
C
      WRITE (6,*) ' COMPUTING C TIMES SF FOR NEW C' SPA02490
C
      WRITE (16,1008) SPA02500
      DO 11000 I=1,200 SPA02510
          DO 10900 J=1,NO SPA02520
              C(J,I)= C( J,I)*SF SPA02530
10900  CONTINUE SPA02540
11000 CONTINUE SPA02550
C

```

```

C      ***** PRE-LOOP PORTION OF KALMAN FILTER           SPA02740
C
C
C      JK=SMODE*2-2                                     SPA02750
C      M2=2*MODE                                       SPA02760
C
C
C      **************************************** SPA02770
C
C      M2=2*MODE                                       SPA02780
C      JP=2*SMODE -1                                  SPA02790
C      JQ=2*EMODE                                     SPA02800
C
C
C      DO 8813 I=1,3                                  SPA02810
C      EX(I)=0.0                                       SPA02820
8813  CONTINUE                                     SPA02830
C
C
C
C      **************************************** SPA02840
C      ***** THIS SECTION COMPUTES THE STATE UPDATE    SPA02850
C      **************************************** SPA02860
C
C      ESS =0.0                                         SPA02870
C      COUNT = 0                                       SPA02880
C      ENERGY = 0.0DO                                  SPA02890
C      TIME = 0.0                                       SPA02900
C      CGN=0.0                                         SPA02910
C      CTG=0.0                                         SPA02920
C
C      ***** SETS LOOP FOR THE ITERATIONS NECESSARY TO SPA02930
C      ***** THE SYSTEM FOR THE NUMBER OF MINUTES SPECIFIED SPA02940
C      **************************************** SPA02950
C      *****      WRITE (6,1008)                         SPA02960
C      *****      WRITE (6,*) '           START STATE UPDATE    ' SPA02970
C      *****      LOOP = INT((MIN*60.0)/SAMPT)          SPA02980
C      *****      PRT= (DBLE(LOOP))/30.0                 SPA02990
C      *****      PRT1=(DBLE(LOOP))/50.00                SPA03000
C      *****      CTT=0.0                                SPA03010
C
C      DO 400 L = 0, LOOP                            SPA03020
C          TIME = DBLE(L)*SAMPT                      SPA03030
C
C          IF(L.EQ.0)THEN                           SPA03040
C              IMPLSE =(1.0D06*SF1)/(DSQRT(SAMPT)) SPA03050
C          ELSE                                     SPA03060
C              IMPLSE = 0.0DO                         SPA03070
C          ENDIF                                    SPA03080
C
C          TO=0.0                                     SPA03090
C
C          ***** RANDOM NUMBER GENERATOR ***** SPA03100
C
C          DO 101 I=1,6                            SPA03110
C              ISEED=MOD(ISEED*JR+KR,MR)           SPA03120
C              RND1=(DBLE(ISEED)+0.5D00)/DBLE(MR) SPA03130
C              ISEED=MOD(ISEED*JR+KR,MR)           SPA03140
C              RND2=(DBLE(ISEED)+0.5D00)/DBLE(MR) SPA03150

```

```

101  RNDM(I)=DSQRT(-2.0*DLOG(RND1))*DCOS(6.2831853D00*RND2) SPA03300
C    CONTINUE SPA03310
C    **** CTT=CTT+1.0 SPA03320
C    *** START OF STATE UPDATE ***
C    *** COMPUTE AX0200 = A0200 X 200 * X0200 SPA03330
C    *** COMPUTE AXH = A*XH SPA03340
C    JK=SMODE*2-2 SPA03350
C    JP=JK+1 SPA03360
C    JQ=2*EMODE SPA03370
C    DO 5015 I=NS,NF SPA03380
C      SUM=0.0 SPA03390
C        DO 5010 K=NS,NF SPA03400
C          SUM=SUM+A(I,K)*X(K) SPA03410
5010    CONTINUE SPA03420
      AX(I)=SUM SPA03430
5015  CONTINUE SPA03440
C    *** COMPUTE WD03 SPA03450
C    WD(1)=PNVARX*RNDM(1)*TO+IMPLSE SPA03460
C    WD(2)=PNVARY*RNDM(2)*TO SPA03470
C    WD(3)=PNVARZ*RNDM(3)*TO SPA03480
C    *** COMPUTE BNWD0200 =BN0200 X 3 * WD03 SPA03490
C    DO 5035 I=NS,NF SPA03500
C      SUM=0.0 SPA03510
C        DO 5030 K=1,3 SPA03520
C          SUM=SUM+BN(I,K)*WD(K) SPA03530
5030    CONTINUE SPA03540
      BNWD(I)=SUM SPA03550
5035  CONTINUE SPA03560
C    *** COMPUTE X0200 =AX0200 + BNWD0200 SPA03570
C    DO 5040 I=NS,NF SPA03580
      X(I)=AX(I) + BNWD(I) SPA03590
      IF (DABS(X(I)).LT. 1.0D-60) THEN SPA03600
        X(I)=1.0D-60 SPA03610
      END IF SPA03620
C    CONTINUE SPA03630
C    **** SPA03640
C    *** START OF KALMAN FILTER ***
C    **** SPA03650

```

```

JK=SMODE*2-2           SPA03860
JP=JK+1                SPA03870
JQ=2*EMODE             SPA03880
M2=2*MODE              SPA03890
C                         SPA03900
JL=JQ+1                SPA03910
DO 8888 I=1,NO          SPA03920
SUM=0.0                 SPA03930
DO 8887 K=JL,NF          SPA03940
SUM=SUM+C(I,K)*X(K)      SPA03950
8887  CONTINUE           SPA03960
EX(I)=SUM*SUM*SAMPT+EX(I) SPA03970
8888  CONTINUE           SPA03980
C                         SPA03990
C                         SPA04000
CGN=CGN+1.0             SPA04010
IF (CTT.EQ.1.0.OR.CGN.GT.PRT) THEN SPA04020
C                         SPA04030
WRITE (16,1008)           SPA04040
WRITE (16,*) 'TIME = ', TIME SPA04050
C                         SPA04060
DO 380 I=JP , JQ          SPA04070
WRITE (16,4500) I,X(I)      SPA04080
380  CONTINUE             SPA04090
4500 FORMAT (' ',2X,'X(' ,I4,')= ',D15.8) SPA04100
C                         SPA04110
CGN=0.0                  SPA04120
END IF                   SPA04130
C                         SPA04140
C                         SPA04150
400  CONTINUE             SPA04160
WRITE (11,*) 'SMODE = ', SMODE SPA04170
WRITE (11,*) 'EMODE = ', EMODE SPA04180
WRITE (11,*) 'SN = ', SN      SPA04190
WRITE (11,*) 'FN = ', FN      SPA04200
C                         SPA04210
JL=JQ+1                SPA04220
DO 9499 I=1,NO          SPA04230
WRITE (11,*) 'EX ',I , '      ', EX(I) SPA04240
9499  CONTINUE           SPA04250
C                         SPA04260
C                         SPA04270
C                         SPA04280
C                         SPA04290
C                         SPA04300
C                         SPA04310
CALL EXCMS ('CLRSCRN')    SPA04320
WRITE (6,1008)             SPA04330
C                         SPA04340
599  STOP                 SPA04350
END                      SPA04360
C                         SPA04370
C                         SPA04380
C                         SPA04390
C                         SPA04400
C                         SPA04410

```

```

C ***** THIS SUBROUTINE COMPUTES THE STATE TRANSITION MATRIX FOR EACH ***** SPA04420
C OF THE 100 MODES ***** SPA04430
C ***** SPA04440
C ***** SPA04450
C ***** SPA04460
C ***** SPA04470
C ***** SPA04480
C ***** SPA04490
C ***** SPA04500
C ***** SPA04510
C ***** SPA04520
C ***** SPA04530
C ***** SPA04540
C ***** SPA04550
C ***** SPA04560
C ***** SPA04570
C ***** SPA04580
C ***** SPA04590
C ***** SPA04600
C ***** SPA04610
C ***** SPA04620
C ***** SPA04630
C ***** SPA04640
C ***** SPA04650
C ***** SPA04660
C ***** SPA04670
C ***** SPA04680
C ***** SPA04690
C ***** SPA04700
C ***** SPA04710
C ***** SPA04720
C ***** SPA04730
C ***** SPA04740
C ***** SPA04750
C ***** SPA04760
C ***** SPA04770
C ***** SPA04780
C ***** SPA04790
C ***** SPA04800
C ***** SPA04810
C ***** SPA04820
C ***** SPA04830
C ***** SPA04840
C ***** SPA04850
C ***** SPA04860
C ***** SPA04870
C ***** SPA04880
C ***** SPA04890
C ***** SPA04900
C ***** SPA04910
C ***** SPA04920
C ***** SPA04930
C ***** SPA04940
C ***** SPA04950
C ***** SPA04960

SUBROUTINE STMTRX(EMODE, SMODE, T, D, PHI, GAMMA, A, B, LAMA, UGVEX, C,
+      ROWN1, ROWN2, ROWN3, BN)
REAL*8 WN, GMA, PHI(2,2,100), GAMMA(2,100), EGT, T, COSW1T, SINW1T
REAL*8 W1, D, A(200,200), B(200,3), C(9,200), BN(200,3)
REAL LAMA(100), UGVEX(684,100)
INTEGER SMODE, R, EMODE, JJ, KK, ROWN1, ROWN2, ROWN3, NN(9), N9, NO
C
C
WRITE (6,*) 'INSIDE STMTRX -- COMPUTE WN, GMA, EFT, W1'
C
C
DO 600 I = 1      ,100
  WN = DBLE(SQRT(LAMA(I)))
  GMA = D*WN/2.0
  EGT = DEXP(-GMA*T)
  W1 = DSQRT((WN**2)-(GMA**2))
  COSW1T = DCOS(W1*T)
  SINW1T = DSIN(W1*T)
C
C
C
C
IF(WN.EQ.0)THEN
  PHI(1,1,I) = EGT*COSW1T
  PHI(1,2,I) = T
  PHI(2,1,I) = 0
  PHI(2,2,I) = EGT*COSW1T
C
C
C
C
  GAMMA(1,I) = 0
  GAMMA(2,I) = 0
ELSE
C
C
C
C
  PHI(1,1,I) = EGT*(COSW1T + (GMA*(W1**(-1)))*SINW1T)
  PHI(1,2,I) = (W1**(-1))*EGT*SINW1T
  PHI(2,1,I) = -(WN**2)*(W1**(-1))*EGT*SINW1T
  PHI(2,2,I) = EGT*(COSW1T - (GMA*(W1**(-1)))*SINW1T)
C
C
C
C
  GAMMA(1,I)=(WN**(-2))*(1.0D0-EGT*COSW1T -EGT*(GMA/W1)*SINW1T)

```

```

C GAMMA(2,I) = (W1**(-1))*EGT*SINW1T SPA04970
C SPA04980
C SPA04990
C SPA05000
C SPA05010
C SPA05020
C SPA05030
C SPA05040
C SPA05050
C SPA05060
C SPA05070
C SPA05080
C SPA05090
C SPA05100
C SPA05110
C SPA05120
C SPA05130
C SPA05140
C SPA05150
C SPA05160
C SPA05170
C SPA05180
C SPA05190
C SPA05200
C SPA05210
C SPA05220
C SPA05230
C SPA05240
C SPA05250
C SPA05260
C SPA05270
C SPA05280
C SPA05290
C *** B MATRIX FOR MULTIPLYING CONTROL TORQUES
C SPA05300
C SPA05310
C SPA05320
C SPA05330
C SPA05340
C SPA05350
C SPA05360
C SPA05370
C SPA05380
C SPA05390
C SPA05400
C SPA05410
C SPA05420
C SPA05430
C SPA05440
C SPA05450
C SPA05460
C SPA05470
C SPA05480
C SPA05490
C SPA05500
C SPA05510
C SPA05520

600 CONTINUE

C WRITE (6,*) 'PHI AND GAMMA COMPUTED'
C WRITE (6,*) ' COMPUTING A, B, BN'
C R = 1
C DO 610 K = 1      ,100

C A(R,R) = PHI(1,1,K)
C A(R,R+1) = PHI(1,2,K)
C A(R+1,R) = PHI(2,1,K)
C A(R+1,R+1) = PHI(2,2,K)

C *** B MATRIX FOR MULTIPLYING CONTROL TORQUES

C B(R,1) = GAMMA(1,K)*DBLE(UGVEX(412,K))
C B(R,2) = GAMMA(1,K)*DBLE(UGVEX(413,K))
C B(R,3) = GAMMA(1,K)*DBLE(UGVEX(414,K))
C B(R+1,1) = GAMMA(2,K)*DBLE(UGVEX(412,K))
C B(R+1,2) = GAMMA(2,K)*DBLE(UGVEX(413,K))
C B(R+1,3) = GAMMA(2,K)*DBLE(UGVEX(414,K))

C *** BN MATRIX FOR MULTIPLYING THE NOISE DISTURBANCES

C BN(R,1)=GAMMA(1,K)*DBLE(UGVEX(ROWN1,K))
C BN(R,2)=GAMMA(1,K)*DBLE(UGVEX(ROWN2,K))
C BN(R,3)=GAMMA(1,K)*DBLE(UGVEX(ROWN3,K))
C BN(R+1,1)=GAMMA(2,K)*DBLE(UGVEX(ROWN1,K))
C BN(R+1,2)=GAMMA(2,K)*DBLE(UGVEX(ROWN2,K))

```

```

C BN(R+1,3)=GAMMA( 2 ,K)*DBLE(UGVEX(ROWN3,K)) SPA05530
C SPA05540
C SPA05550
C SPA05560
C SPA05570
C SPA05580
C SPA05590
C SPA05600
C R = R+2 SPA05610
610 CONTINUE SPA05620
C WRITE (6,*) 'A, B, BN COMPUTED' SPA05630
C WRITE (6,*) 'COMPUTING C' SPA05640
C SPA05650
C SPA05660
C SPA05670
C SPA05680
C *** C MATRIX PRODUCTION ***
C SPA05690
NO=3 SPA05700
NN(1)=418 SPA05710
NN(2)=419 SPA05720
NN(3)=420 SPA05730
C SPA05740
C SPA05750
C SPA05760
C SPA05770
C SPA05780
C SPA05790
C JJ=-1 SPA05800
DO 640 I=1,100 SPA05810
JJ=JJ+1 SPA05820
C SPA05830
DO 9127 K=1,NO SPA05840
C SPA05850
KK=I+JJ SPA05860
C SPA05870
N9=NN(K) SPA05880
C SPA05890
C(K,KK) = DBLE(UGVEX(N9,I)) SPA05900
C SPA05910
KK=KK+1 SPA05920
C SPA05930
C C(K,KK)=0.0 SPA05940
9127 CONTINUE SPA05950
640 CONTINUE SPA05960
C SPA05970
C SPA05980
C SPA05990
C RETURN SPA06000
END SPA06010

```

LIST OF REFERENCES

1. Rodriguez, G., *Control Technology Development*, Large Space Systems Technology - 1980, NASA Conference Publication 2168, Vol. I, pp. 49-63, November 18-20, 1980.
2. Thomson, William T., *Theory of Vibration with Applications*, Prentice-Hall, Inc., Englewood Cliffs, NJ, 1988.
3. Preston, William J. , *Effects of Reduced Order Modeling on the Control of a Large Space Structure*, Masters Thesis, Naval Postgraduate School, CA, September 1988.
4. Selby, Samuel M., *Standard Mathematical Tables*, The Chemical Rubber Co., Cleveland, OH, 1972.
5. Mendel, Jerry M., *Discrete Techniques of Parameter Estimation*, Marcel Dekker, Inc., NY, 1973.
6. Nishimura, T., "On the a priori Information in Sequential Estimation Problems," *Kalman Filtering: Theory and Application*, p. 138-145, Edited by: Sorenson, H. W., IEEE Press, Inc., New York, 1985.
7. Franklin, Gene F. and Powell, David J., *Digital Control of Dynamic Systems*, Addison-Wesley Company, Reading, MA, 1980.

INITIAL DISTRIBUTION LIST

	No. Copies
1. Defense Technical Information Center Cameron Station Alexandria, VA 22304-6145	2
2. Library, Code 0142 Naval Postgraduate School Monterey, CA 93943-5002	2
3. Director Naval Research Laboratory Washington, D.C. 20375	1
4. Commander Naval Space Command Attn: Code N3 Dahlgren, VA 22448-5170	1
5. Chairman, Code 62 Department of Electrical and Computer Engineering Naval Postgraduate School Monterey, CA 93943	1
6. Prof. Jeff B. Burl, Code 62BL Department of Electrical and Computer Engineering Naval Postgraduate School Monterey, CA 93943	1
7. Prof. H. A. Titus, Code 62TS Department of Electrical and Computer Engineering Naval Postgraduate School Monterey, CA 93943	1
8. Major W. J. Preston 7 Cary Place Fredricksburg, VA 22405	1
9. Lcdr. Bruce M. Jackson c/o Col. Jose A. Aybar (USA-Ret) 22577 Veronica Drive Salinas, CA 93943	2

Thesis
J2125 Jackson
c.1 Utilization of a Kal-
man observer with large
space structures.

15 APR 92

37043

Thesis
J2125 Jackson
c.1 Utilization of a Kal-
man observer with large
space structures.



thesJ2125
Utilization of a Kalman observer with la



3 2768 000 81616 9

DUDLEY KNOX LIBRARY